Lock Selection in Practice

Abdelsalam Shanneb and John Potter

Programming Languages and Compilers Group
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052, Australia
{shanneba,potter}@cse.unsw.edu.au

Technical Report

UNSW-CSE-TR-605

Abstract

This report is a continuation of our last report (UNSW-CSE-TR-604). Here, we present our algorithm for offering lock choices. We show all the details involved in making lock choices for programmers. We offer different approaches to lock selection; we propose top-down and bottom-up strategies, as well as, additive and subtractive techniques. These lock choices are based on the previously established Galois connection between the layers of a composite. We also present several examples that demonstrate our prototype tool for lock selection, the output of which is in Appendix A.

1 Introduction

The main goal of this report is to demonstrate the feasibility of automated support for the selection of locks in composite systems. We identify various strategies for lock selection: the subtractive approach and the additive approach. We also introduce the two directions, top-down and bottom-up, and show how they are incorporated with the additive and subtractive approaches. We also present some formal properties to support our algorithm for efficiently calculating the minimal lock choices.

This report should be read together with the previous report (UNSW-CSE-TR-604) on which it relies on for background discussion and associated references, omitted here. More importantly the previous report summarises our model for composite objects with internal exclusion requirements R, and external concurrency potential P, together with the characterisation of safe and non-redundant locking for the composite. It finalises the outward propagation of exclusion requirement, $R^{\triangleleft}$, and inward propagation of concurrency potential, $P^{\triangleright}$, given the dependency relation between components. Most of this work has also been published in [SP05, SPN05]. Finally, the previous report investigated properties of the fixpoint lattice for the Galois connection formed by $\triangleright$ and $\triangleleft$, and discussed its relevance for lock selection. We rely heavily on this previous report as we now explore strategies for lock selection.

The remainder of this report is organised as follows. Section 2 introduces general strategies for lock selection within a composite object. We walk through examples of the top-down approach for lock selection in Section 3. Section 4 describes the bottom-up approach and explains the issues associated with approach. Section 5 introduces our model for collapsible edges; a way for reducing the size of the lattice of fixpoints. This model leads to developing a more efficient algorithm for calculating fixpoints and stepping through a lattice. This algorithm with its underlying properties and definitions are detailed in Section 6. Section 7 concludes the report.

2 Lock Selection Strategies

2.1 Top-down vs Bottom-up

At any stage of lock selection we have the choice of achieving safety for a particular component by applying a lock on that component, or by delegating the exclusion requirements, partially or totally, to another component. In our model, we assume that all the exclusion requirements derive from leaf components in a hierarchical structure; consequently we can either adopt a top-down or a bottom-up lock selection strategy.

The only difference between the top-down and bottom-up approaches is the direction where the exclusion requirements are propagated. In the top-down approach, at any component, we select a lock for an outer component, as in Figure 1. If some of the composite exclusion requirements are not met at that component, then the remaining requirement should be delegated down to the lower components where these missing requirements are either met or delegated again even lower, and so on.

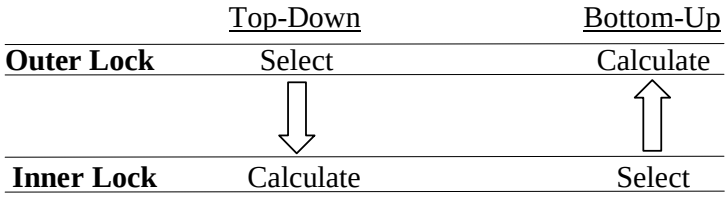


Figure 1: Top-Down vs Bottom-Up

The bottom-up approach basically starts at the bottom level components where it considers all exclusion requirements at that level and then offers a user several choices for lock selection. A user may decide not to select any of the locks or select part of the lock being offered, in this case the remaining exclusion requirement propagate to upper levels where the remaining exclusion requirement must be met. The decision for meeting the requirements or delegating it to another component is based on the user choice of the locks being offered at that component. When choosing between the two approaches, we prefer the top-down

approach as we explain later in Section 4.1.

2.2 Additive vs Subtractive

With both top-down and bottom-up approaches we can either add or remove locks on a particular component as suggested by Figure 2.

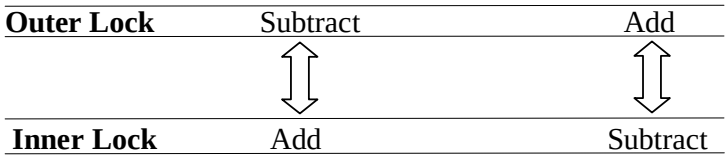


Figure 2: Additive vs Subtractive

So we have either an additive or a subtractive method. With a subtractive top-down approach, we remove locks at the outer level, thereby requiring extra locks to be added at the inner level. The starting configuration is with coarse-grain locking at the outermost level. After locks are removed at one level (and added at the next level) we move to the next level, and consider further removals there. The effect is to transform a coarse-grain lock into successively finer-grain locks, always maintaining overall system safety with no redundant locks.

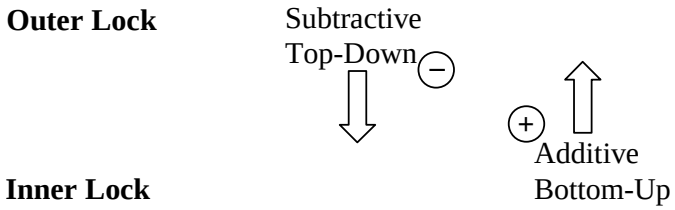


Figure 3: Subtractive Top-Down vs Additive Bottom-Up

Assuming we restrict attention to non-redundant locks, the results of [SP06] indicate that we can achieve the same effect by making additive selection bottom-up, as suggested by Figure 3. With this selection policy we start with the same initial lock configuration – with all locks at the outermost level. But then we add locks from the innermost level, thereby reducing the outer requirements right up to the outermost level. We can then

proceed bottom-up adding locks to meet the (possibly reduced) requirements at successive levels, thereby reducing the outermost lock.

The other complementary pair of selection strategies are additive top-down/subtractive bottom-up, as suggested by Figure 4. The starting configuration for these strategies is the fine-grained approach where all locking requirements are met at the innermost level. Otherwise these strategies work in an analogous way as the subtractive top-down/additive bottom-up strategies.

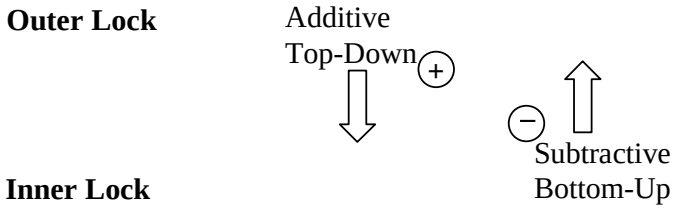


Figure 4: Additive Top-Down vs Subtractive Bottom-Up

There are two benefits of the subtractive approach. First lock selection only affects the next level, so the lock updating is localised. Perhaps more significantly, the subtractive approach allows selection choices to be partitioned. Every possible exclusion pair determines a unique selection. Every selection can therefore be represented by any one of its pairs. This is not true for the additive approach; either bottom-up or top-down can produce overlapping sets of increments, with no guarantee that every possible lock selection can be identified by a single pair within that selection.

3 Top-down Approach

In this section we start with two examples that demonstrate the subtractive method for lock selection, then we present a third example that demonstrates the additive method. Refer to the previous report and papers for the explanation of the notation and calculations used, and explanation of the Galois connection and its fixpoint lattice.

3.1 Subtractive Method - Example 1

Figure 5 depicts a uses relation graph that involves the two components A and B.

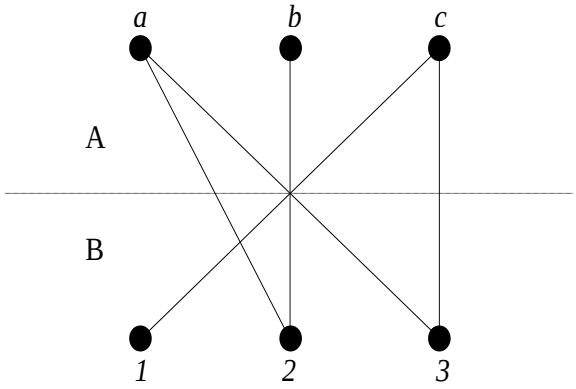


Figure 5: Usage Graph for Example 1

Given a lock requirement on B, $R_B = 1 \times \bar{2} \times 3$, and external potential concurrency on A, $P_A = \overline{abc}$, Figure 6 shows the results after applying the outward and inward mappings.

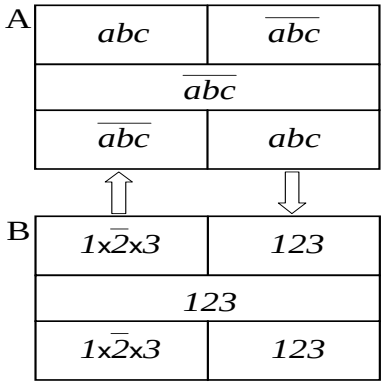


Figure 6: Initial Configuration of Example 1 (Step 0)

The top-down subtractive approach starts by applying a lock that meets the requirement at the top component, in this case the lock $\overline{abc}$ satisfies the exclusion requirement propagated from the lower components, however, with this coarse setting, no form of concurrency is achieved, so we start to decrease the lock in order to allow more concurrency. The process of relaxing the lock at the top level relies on the fixpoint lattice as shown in Figure 7.

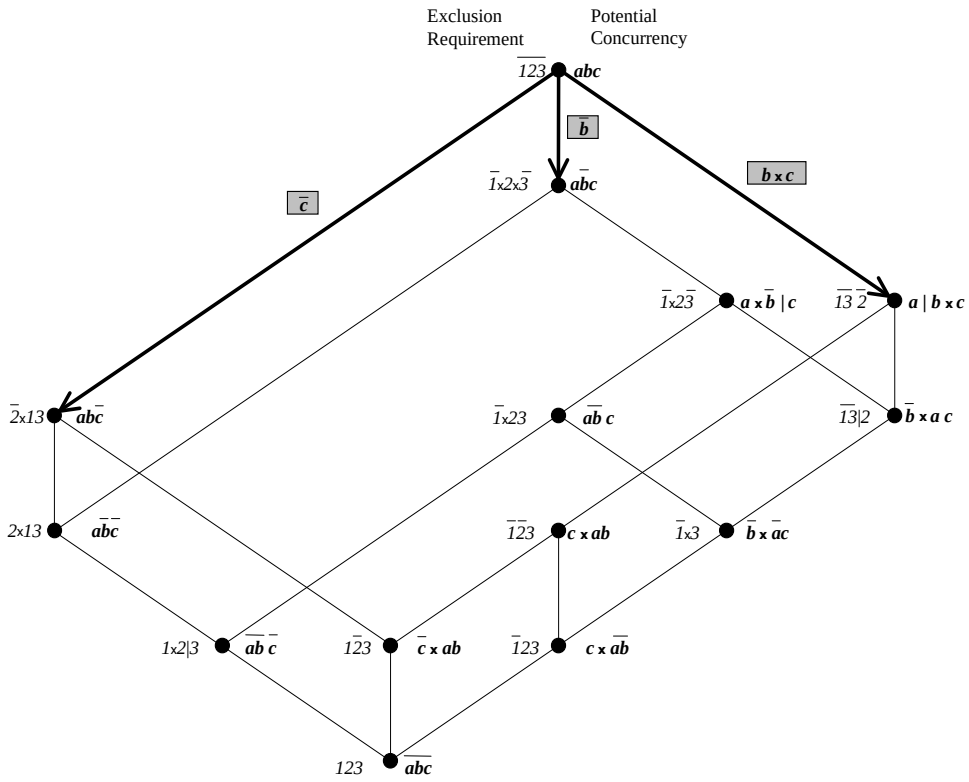


Figure 7: Fixpoint Lattice for Example 1 (Step 1 choices)

Looking at the lattice, we notice that the closure of the given exclusion requirement $R_B = 1 \times \bar{2} \times 3$ is at the top of the lattice $R_B^{\triangleleft\triangleright} = \overline{123}$ which corresponds to the potential concurrency $R_B^{\triangleleft} = abc$. In the lattice, this fixpoint covers three R_B/P_A fixpoints: (i) $\bar{1} \times 2 \times \bar{3}/a\bar{b}c$, (ii) $\bar{2} \times 13/ab\bar{c}$, (iii) $\overline{13}\bar{2}/a|b \times c$. These correspond (respectively) to the addition of concurrency potential pairs within A : (i) $\bar{b}$ (ii) $\bar{c}$ (iii) $b \times c$ as shown on the corresponding edges. Adding any other potential within A reduces to a lower fixpoint, so these three choices are the minimal ones at this stage. Adding potential within A is achieved by subtracting from the lock at A. So these three choices are presented to the user as their first choice for locks to be subtracted at A, as shown by the highlighted cells in Figure 8.

Assuming the user chooses $\bar{b}$, that is, the lock $\bar{b}$ is to removed from the lock at the top component, this will take us to the fixpoint $\bar{1} \times 2 \times \bar{3}/a\bar{b}c$, immediately below our original fixpoint. Now, for component B, this corresponds to a potential of $1\bar{2}3$. The requirement

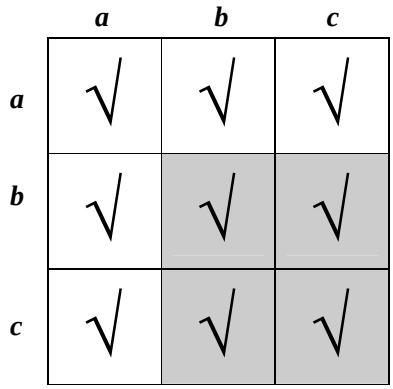


Figure 8: Initial Configuration for Example 1 (Step 1 choices)

on B is still $1 \times \bar{2} \times 3$, and so we now (automatically) satisfy this by choosing the local lock L_B as:

$$\begin{aligned} L_B &= R_B \cap P_B \\ &= 1 \times \bar{2} \times 3 \cap 1\bar{2}3 \\ &= 1\bar{2}3 \end{aligned}$$

Note that the missing requirement is

$$\begin{aligned} R'_B &= R_B - L_B \\ &= 1 \times 2 \times 3 \end{aligned}$$

whose closure is $R'^{\triangleleft\triangleright}_B = \bar{1} \times 2 \times \bar{3}$, which is the selected fixpoint, as expected.

Figure 9 shows the outward and inward mapping after the removal of the lock $\bar{b}$ from A. Notice the addition of the lock $\bar{2}$ within the B component to maintain safety. After the lock $\bar{b}$ is selected, the choice matrix shows different lock choices. The current fixpoint covers two fixpoints: (i) $\bar{1} \times 2\bar{3}/a \times \bar{b}|c$ and (ii) $2 \times 13/a\bar{b}\bar{c}$ as shown in the fixpoint lattice of Figure 10.

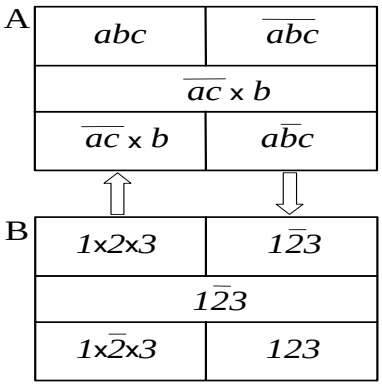


Figure 9: Up and Down for Example 1 (Step 1)

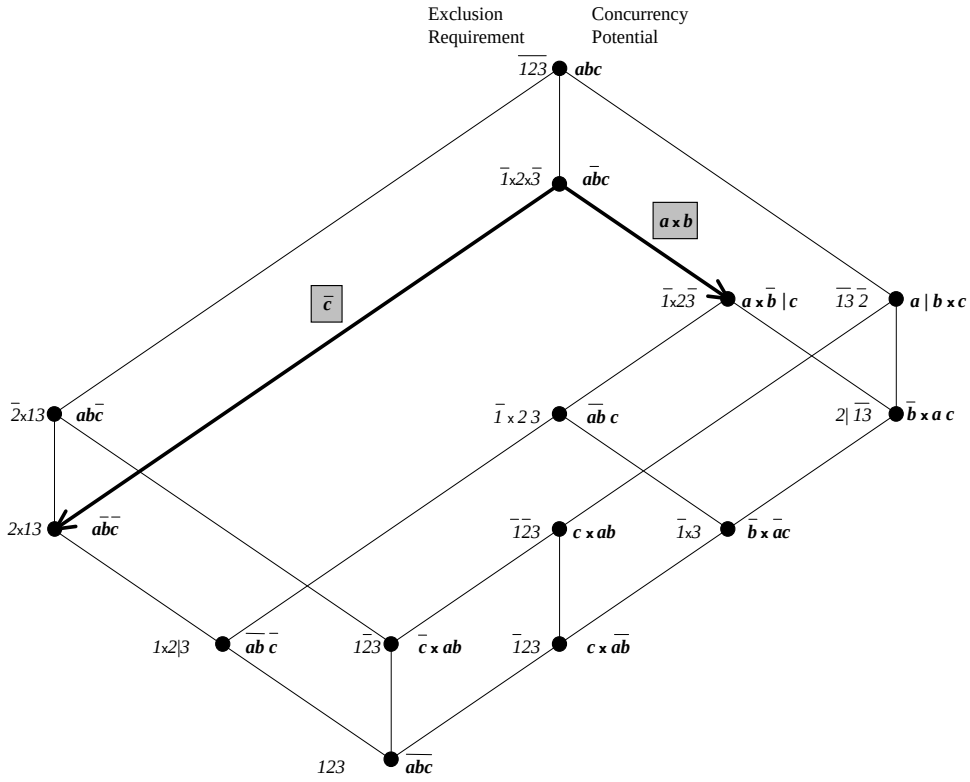


Figure 10: Fixpoint Lattice for Example 1 (Step 2 choices)

The corresponding locks offered for subtractions at A are (i) $a \times b$ and (ii) $\bar{c}$, as shown in Figure 11. Note that $a \times b$ is a new choice, whereas $\bar{c}$ was offered at the previous stage. The other previous choice was $b \times c$, but that choice now leads to fixpoint $2|\overline{13}/\bar{b} \times ac$ corresponding to a lock subtraction of $b \times ac$, which is no longer minimal. So the choice of $b \times c$ does not appear in Figure 11. Suppose that now $\bar{c}$ is selected for subtraction, hence moving to the fixpoint $2 \times 13/a\bar{b}\bar{c}$. Figure 12 shows the resulting configuration. Note that L_B does not block $\bar{1}$ or $\bar{3}$ because they are not part of B's exclusion requirement. From the current fixpoint there is only one minimal subtraction, $\bar{a} \times b$, leading to the fixpoint $1 \times 2|3/\overline{ab}\bar{c}$ as shown in the lattice of Figure 13.

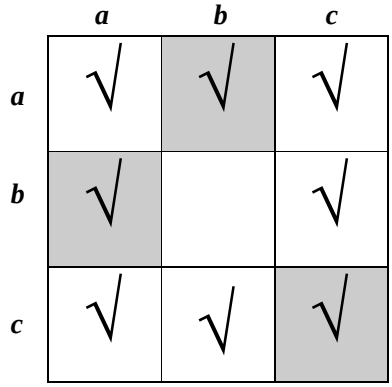


Figure 11: Choice Matrix for Example 1 (Step 2 choices)

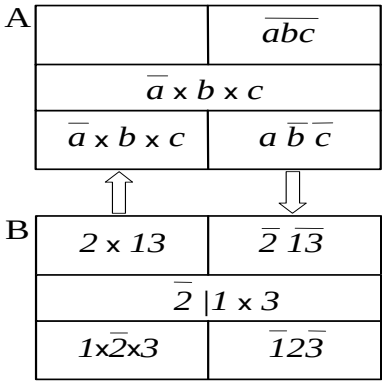


Figure 12: Up and Down for Example 1 (Step 2)

Note that this choice comprises two exclusion pairs. As shown in Figure 14, this choice is presented as if each pair can be subtracted independently. However, if either pair is selected

for subtraction, the other pair is automatically selected with it. This will be reflected in the next matrix. If users are not happy with the automatic subtraction, they can simply undo their choice.

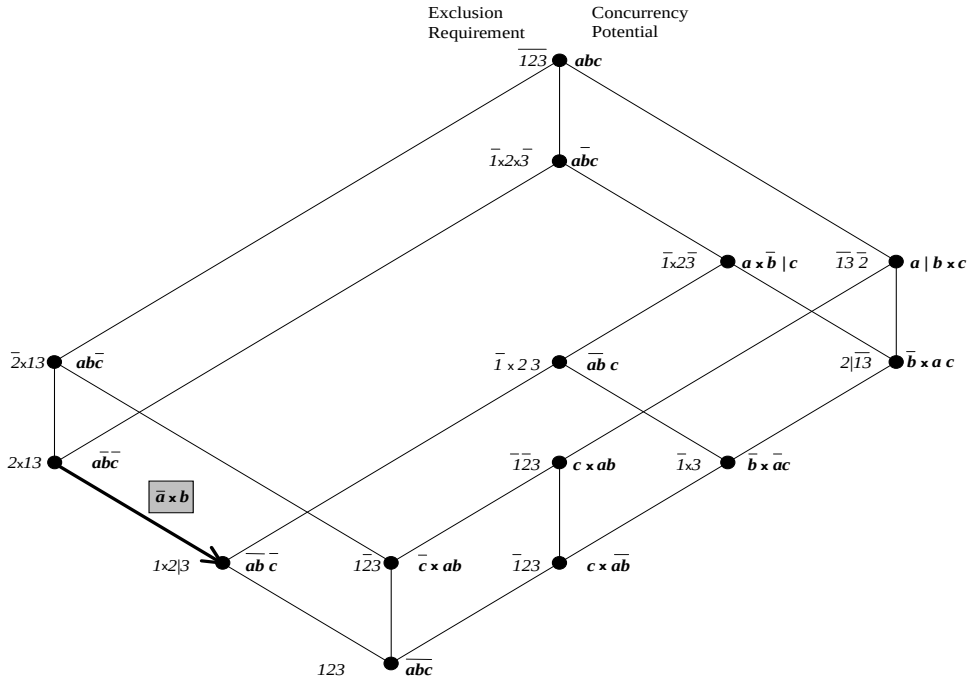


Figure 13: Fixpoint Lattice for Example 1 (Step 3 choices)

	a	b	c
a	√	√	√
b	√		√
c	√	√	

Figure 14: Choice Matrix for Example 1 (Step 3 choices)

Figure 15 shows the result after selecting $\bar{a}$ (with $a \times b$), and Figure 16 depicts the next choice, which we assume is taken. In practice a user could finish with a component at any stage and then move to a lower level component, removing any current locks from there. Figure 17 shows the selected path through the fixpoint lattice.

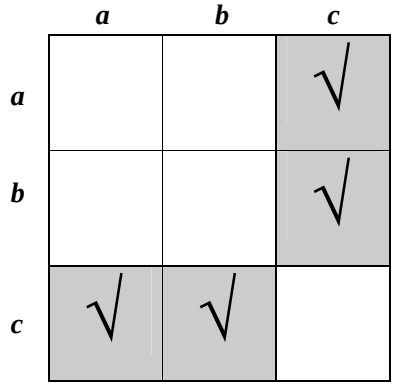


Figure 15: Choice Matrix for Example1 (Step 4 choices)

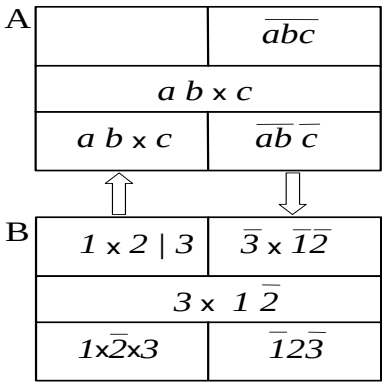


Figure 16: Up and Down for Example 1 (Step 3)

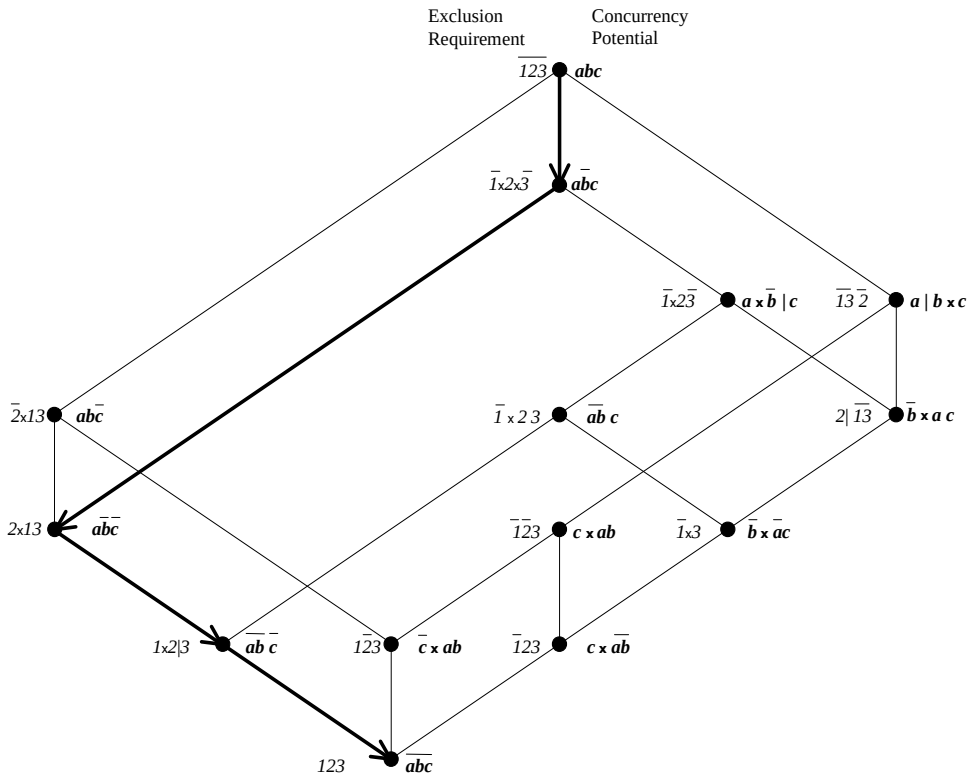


Figure 17: Lock Selections

3.2 Subtractive Method - Example 2

This is another example of the subtractive approach. In this example we demonstrate the lattice reduction property; we show how a lattice may get reduced by collapsing some of its edges if they do not overlap with the exclusion requirement. We use the usage pattern shown in Figure 18. Notice the usage dependency for methods b and 2; method 2 is used by all methods in A while method b uses all methods in B. Based on the usage pattern, Figure 19 depicts the corresponding full fixpoint lattice.

Figure 20 shows the previous lattice with labelled edges. Each label shows a pair of values that corresponds to either a subtracted exclusion requirement or an added concurrency potential in the form R_B/P_A on each the lattice edges. Each edge represents the difference between the two fixpoints on the sides of that edge.

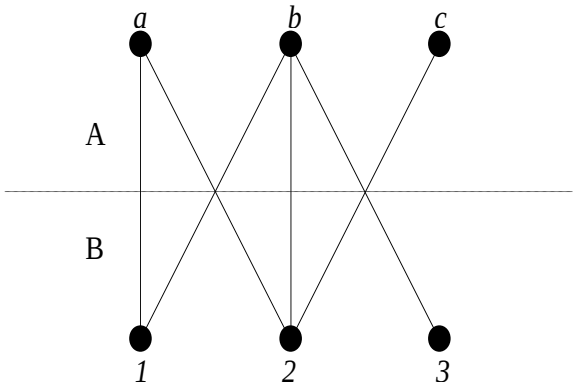


Figure 18: Usage Graph for Example 2

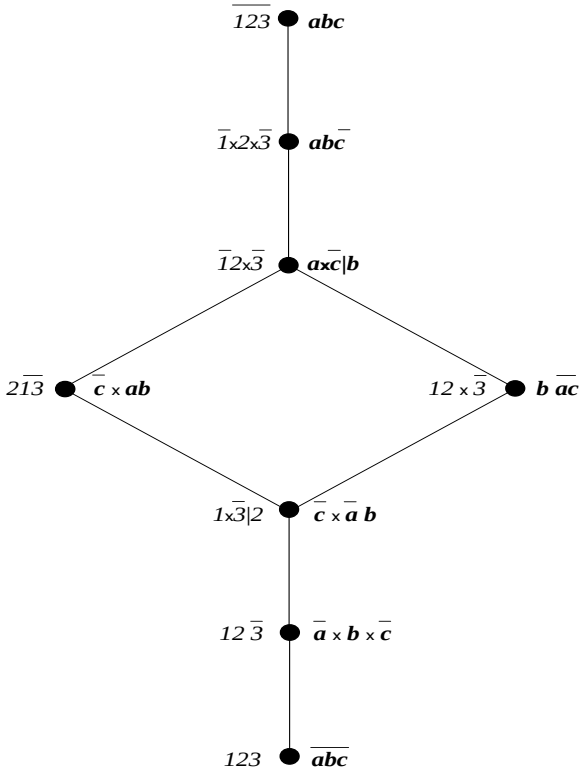


Figure 19: Full Fixpoint Lattice

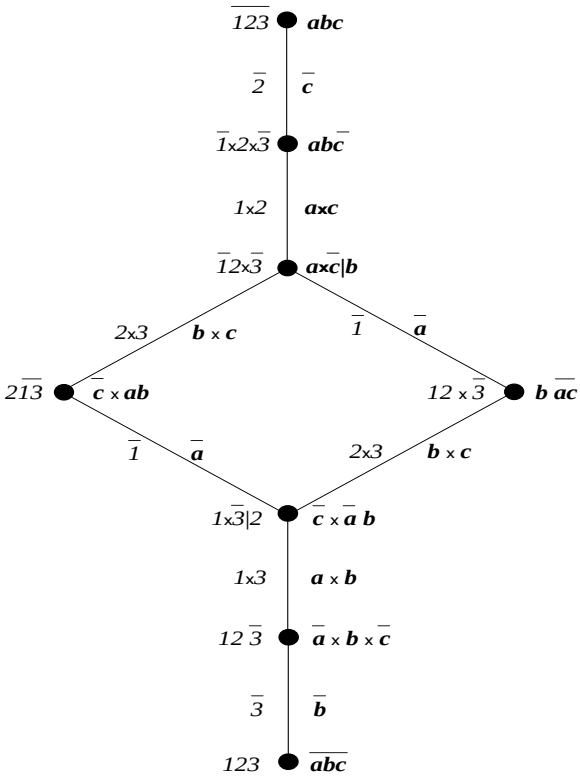


Figure 20: Lattice with Labelled Edges

Now, for the given internal exclusion requirement $R_B = 1 \times \bar{2} \times 3$, we note that those edges labelled with $\bar{1}$, having no part in the requirement; we can join fixpoints which are not distinguished by the requirements, as shown in Figure 21. The reduced lattice (Figure 22) forms a linear order for lock choices, that is, at any stage, the lock choice being offered is either taken or left by the user.

Starting with the exclusion requirement $R_B = 1 \times \bar{2} \times 3$, Figure 23 shows the stages of the lock selection process. Initially, the given exclusion requirement again has its closure at the top of the generated fixpoint lattice, in the dual fixpoint exclusion requirement/potential concurrency $\overline{123}/abc$. The initial lock configuration shown in (a) of Figure 23 indicates a mutex on the top component. Notice the fully locked choice matrix shown in (1). Tracing the fixpoint lattice by following the edges as we go down from one fixpoint to another, the next fixpoint offers the next minimal change by adding concurrency pair $\bar{c}$ within component A. This is achieved by subtracting $\bar{c}$ from the current lock on A, so the user is offered only

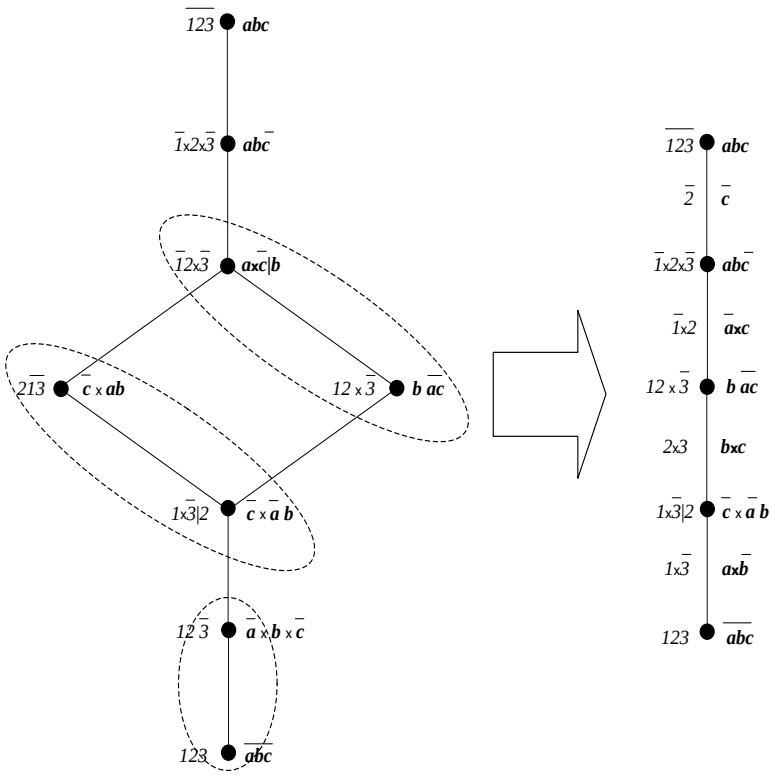


Figure 21: Reduced Lattice

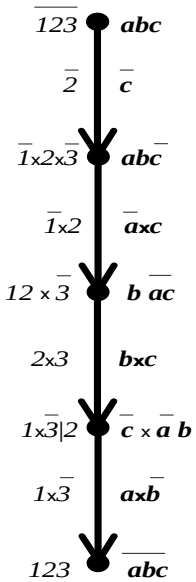


Figure 22: Lock Selections

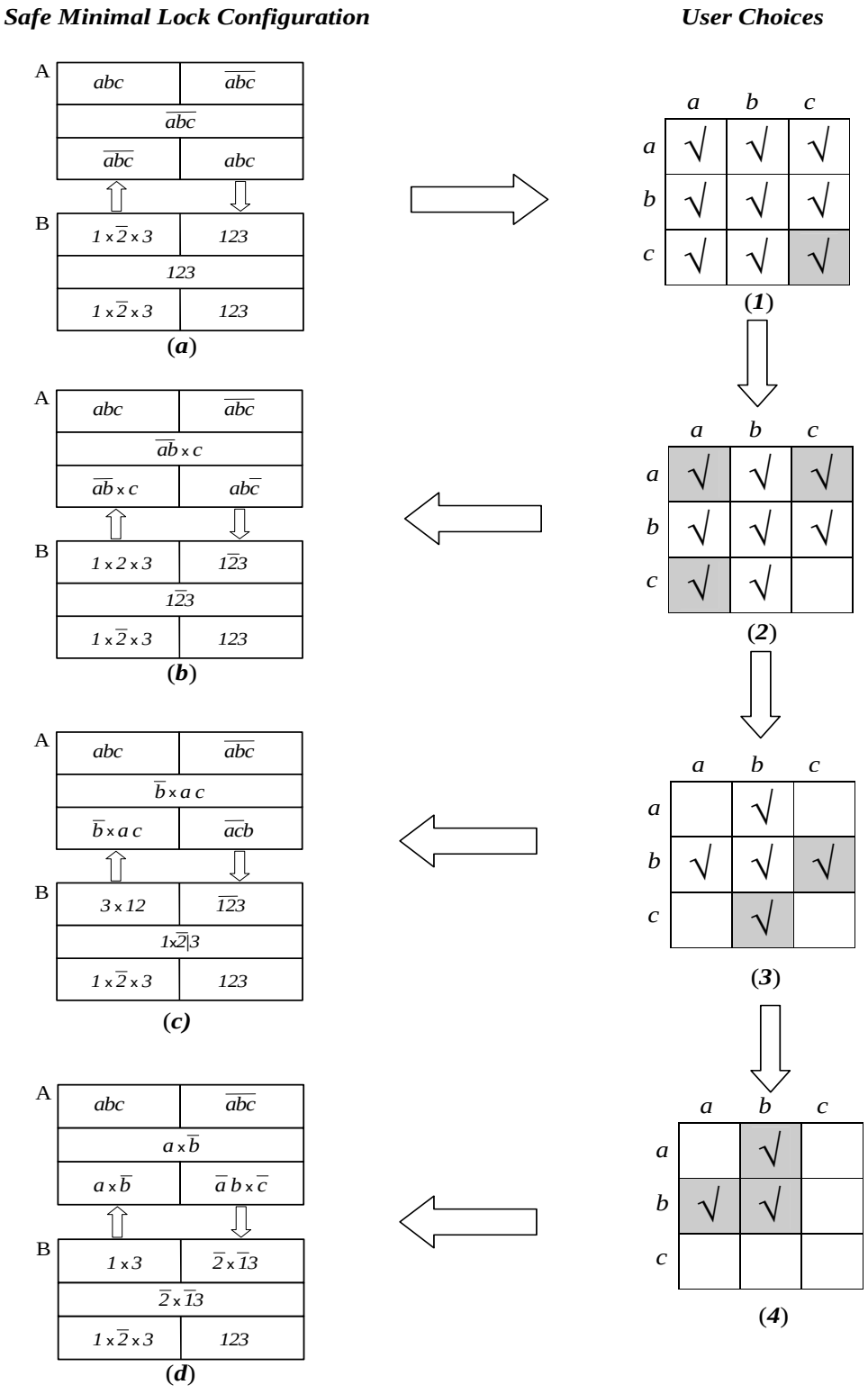


Figure 23: Example 2 - Subtractive Approach - Lock Selection Stages

one choice: subtract the lock $\bar{c}$ as shown in matrix (1), or not. Having that lock subtracted from component A, the selection process takes us to the next fixpoint $\bar{1} \times 2 \times \bar{3}/ab\bar{c}$. Notice the change in lock configuration in (b) which reflects the previous selection of subtracting the lock $\bar{c}$, component B now has to in increase its local lock L_B as:

$$\begin{aligned} L_B &= R_B \cap P_B \\ &= 1 \times \bar{2} \times 3 \cap 1\bar{2}3 \\ &= 1\bar{2}3 \end{aligned}$$

At fixpoint $\bar{1} \times 2 \times \bar{3}/ab\bar{c}$, we again have only one choice to take which leads to fixpoint $12 \times \bar{3}/b\overline{ac}$. The lock choice being offered for subtraction at this stage is $\bar{a} \times c$. Adding either $\bar{a}$ or $a \times c$ to the concurrency potential on A (by subtracting them from the lock) implies adding the other. This leads to the configuration (c). Again, the local lock L_B is increased according to the relation:

$$\begin{aligned} L_B &= R_B \cap P_B \\ &= 1 \times \bar{2} \times 3 \cap \overline{12}3 \\ &= 1 \times \bar{2}|3 \end{aligned}$$

The next choice for lock subtraction is $b \times c$ as shown in choice matrix (3). This new choice leads to fixpoint $1 \times \bar{3}/\bar{c} \times \bar{a}b$. L_B is again determined as:

$$\begin{aligned} L_B &= R_B \cap P_B \\ &= 1 \times \bar{2} \times 3 \cap \bar{2} \times \bar{1}3 \\ &= \bar{2} \times 13 \end{aligned}$$

The final choice is to subtract the remaining of L_A, that is, $a \times \bar{b}$ as shown in matrix (4). Interestingly the only choice in this example has been to take it or leave it, as indicated by the non-branching nature of Figure 22.

3.3 Additive Method Example

The top-down additive approach starts with an empty lock at the top component, and gradually adds appropriate locks until requirements are met. At any stage, each leaf component should provide a lock to guarantee safety. In this example we use the same usage graph from Example 1 of Section 3.1 (Figure 24 repeats Figure 5)

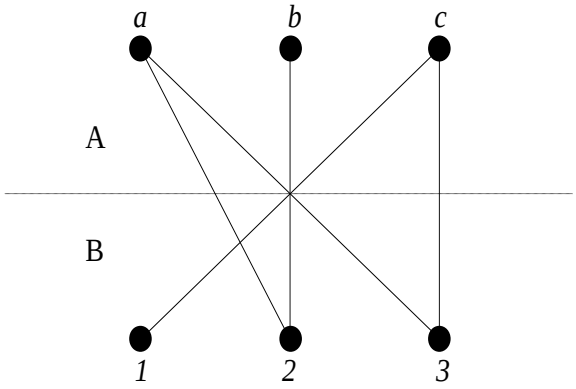


Figure 24: Usage Graph

We also use the exclusion requirement from the previous example: $R_B = 1 \times \bar{2} \times 3$. Figure 25 shows the relevant fixpoint lattice which is identical to the lattice from Example 1

except now we display all edge values. Those edges that are labelled with pairs that do not overlap with the exclusion requirements are highlighted in Figure 25. By collapsing these edges we produce the reduced fixpoint lattice of Figure 26.

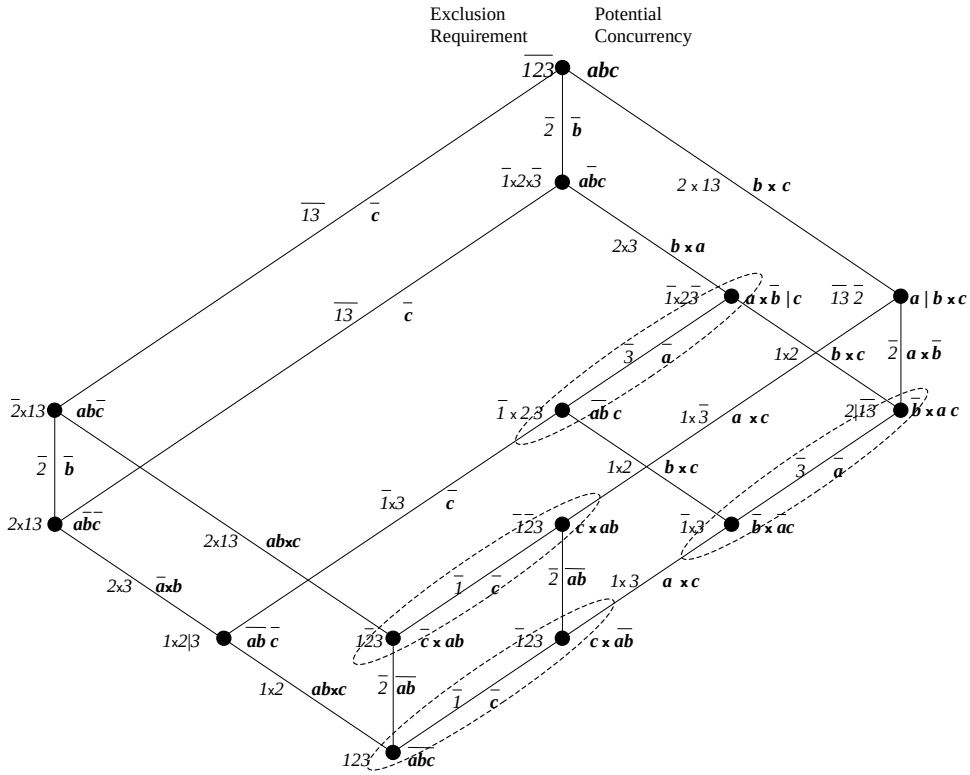


Figure 25: Full Lattice with Collapsible Edges

Working from this reduced lattice, Figure 27 shows the different stages for lock configurations and lock choices for adding to A. Notice that at the initial stage (a) the full internal potential concurrency $\overline{abc}$ at component A, arises from the lack of locks at component A. Notice we also presume internal safety from inner level locking. The current potential concurrency corresponds to a fixpoint at the bottom of the fixpoint lattice; the fixpoint $123/\overline{abc}$ is our starting point with three choices that lead to fixpoints (i) $\bar{1}\times 3|2/\bar{b}\times\bar{a}c$, (ii) $1\bar{2}3/\bar{c}\times ab$, and (iii) $1\times 2|3/\overline{ab}\bar{c}$. These fixpoints correspond to adding the locks (i) $a\times\bar{c}$, (ii) $\overline{ab}$ and (iii) $ab\times c$ respectively as shown in Figure 28.

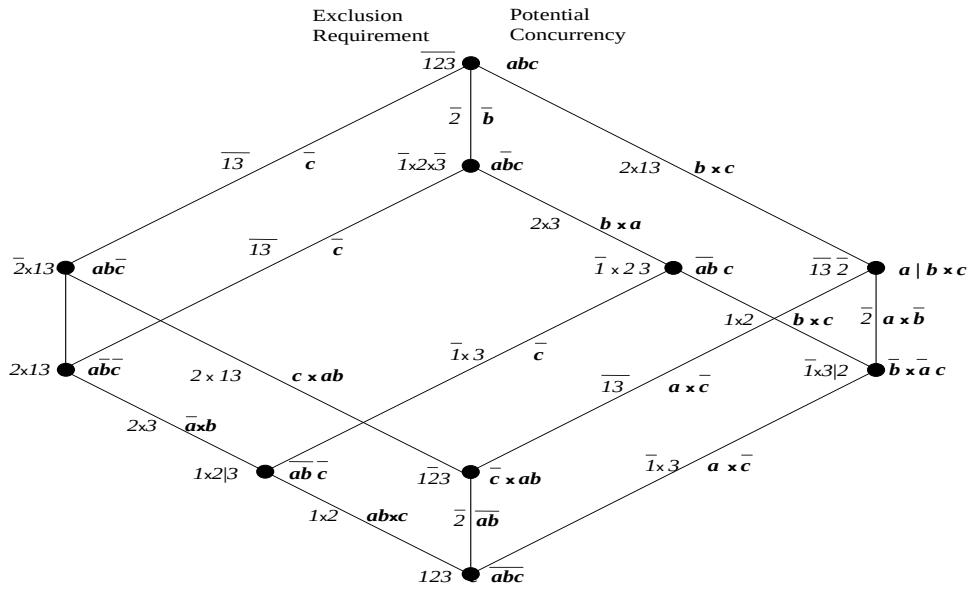


Figure 26: Reduced Fixpoint Lattice

Basically, these lock choices cover all pair combinations. However, the question-mark in choice matrix (1) indicates redundancy or overlap between the choices $a \times \bar{c}$ and $ab \times c$, that is, the pair $a \times c$ occurs in the two lock choices. A problem may arises if the user decided to choose $a \times c$ first; this pair does not uniquely determine the fixpoint. This implies a more complicated interaction with the user to resolve the ambiguity; we do not address this further here. Instead, assume the user has decided to add the lock $\bar{c}$; this automatically adds the lock $a \times c$. Notice L_B reduces as L_A increases, as depicted in lock configuration (b).

The last lock choice takes us to the fixpoint $\bar{1} \times 3|2/\bar{b} \times \bar{a}c$ where from there we have two choices as shown in Figure 29. Matrix 2 shows the available choices; $\bar{a} \times \bar{b}$ and $b \times c$. In this scenario, the user has selected to lock $\bar{a} \times \bar{b}$ as shown in matrix 3, this action produces the configuration depicted in c, and moves the process to fixpoint $\overline{13}\bar{2}/a|b \times c$, where we only have one last lock to offer, that is, $b \times c$. Matrix 4 shows the last selection taken and accordingly the lock configuration in d. Figure 30 depicts the total path taken for the sequence of lock selections.

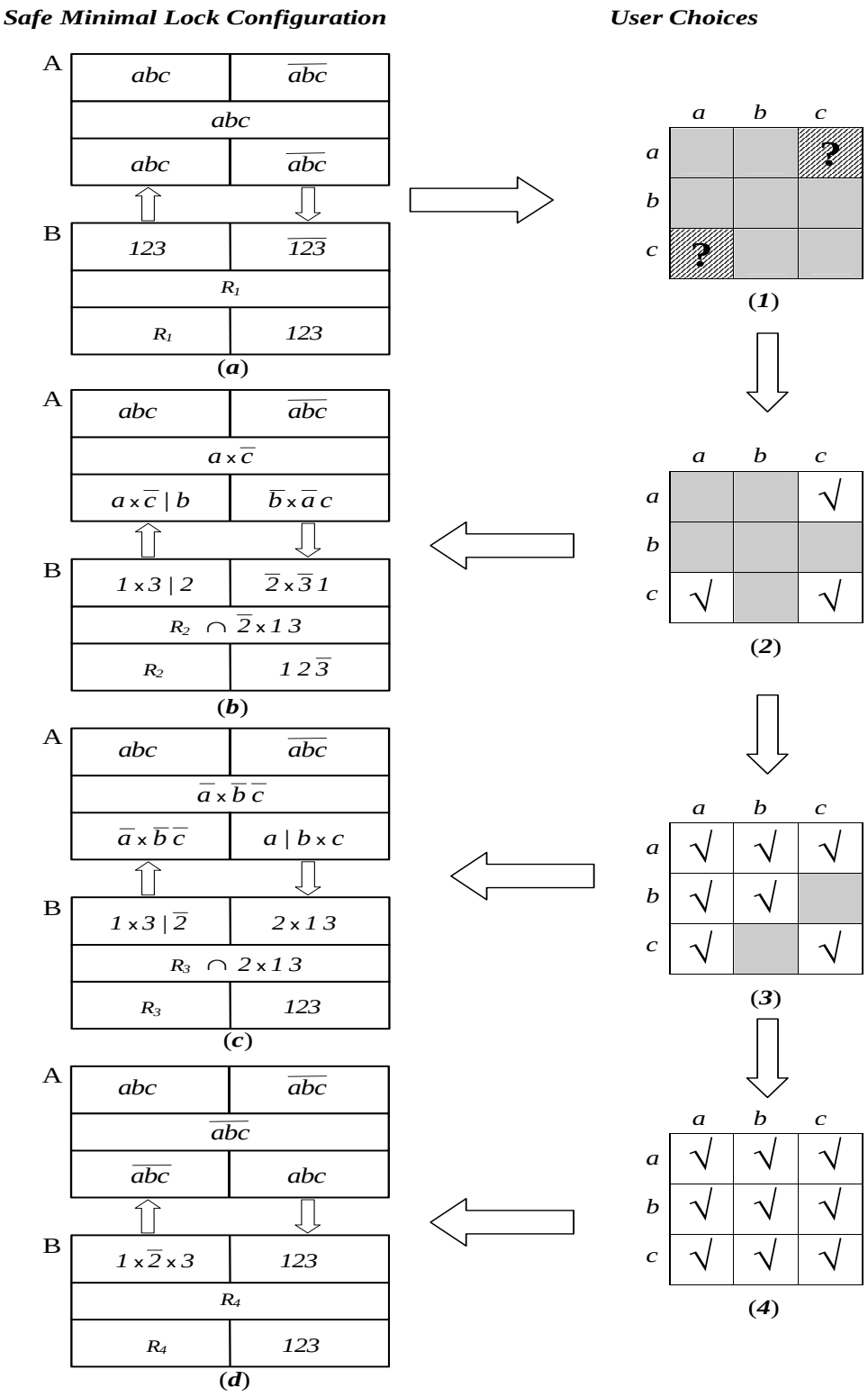


Figure 27: Example of Additive Method - Lock Selection Stages

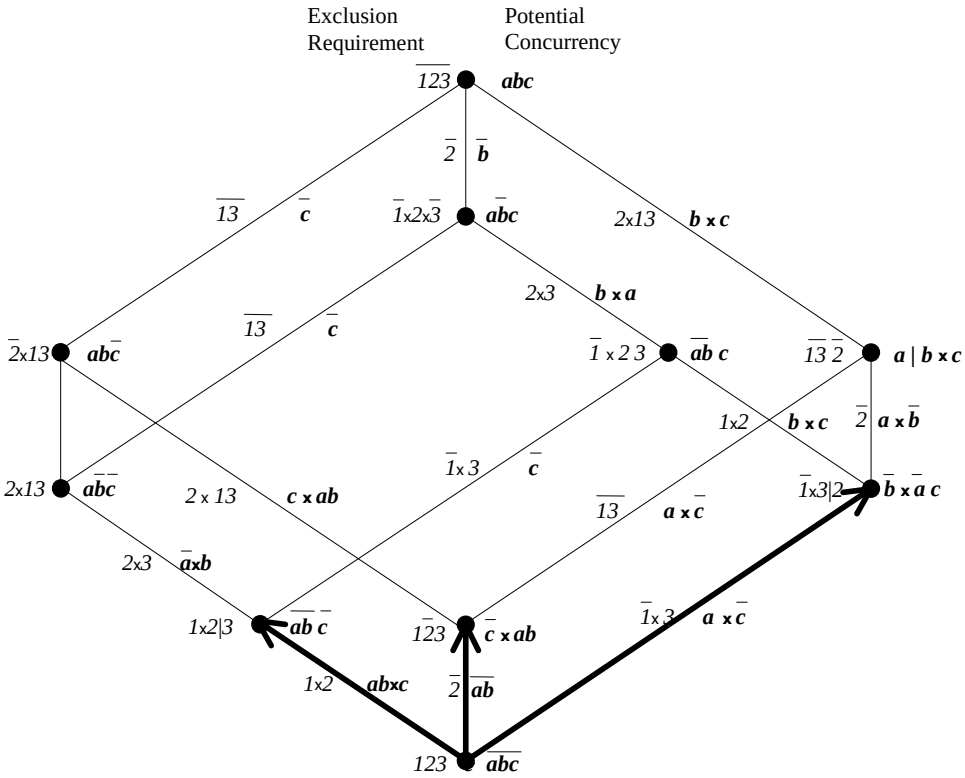


Figure 28: Fixpoint Lattice 1

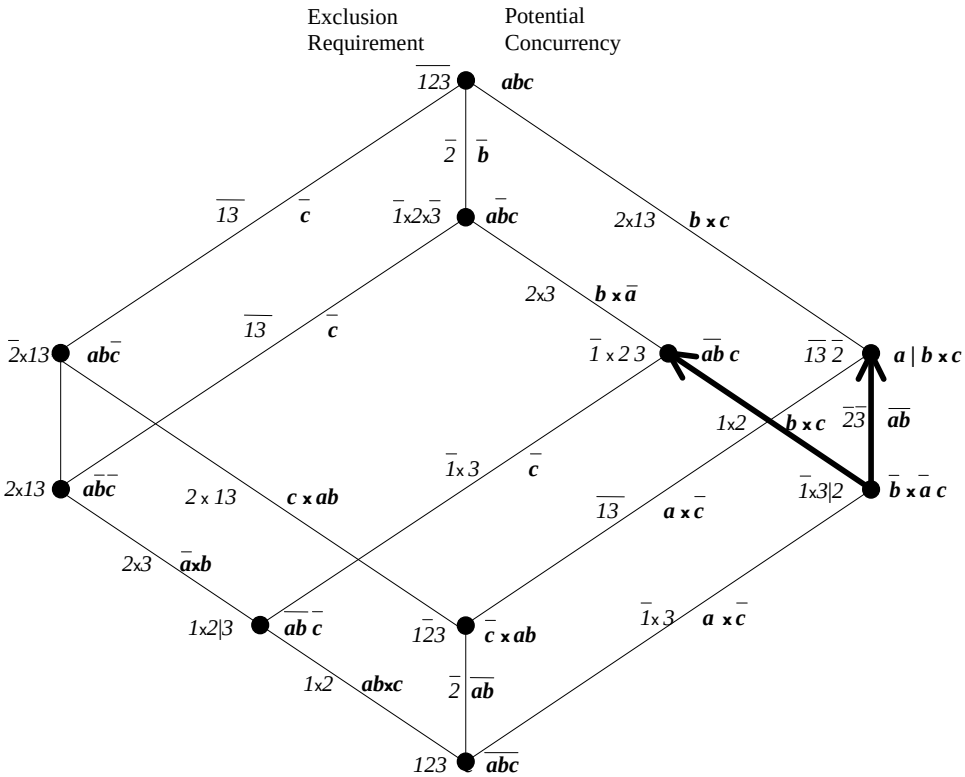


Figure 29: Fixpoint Lattice 2

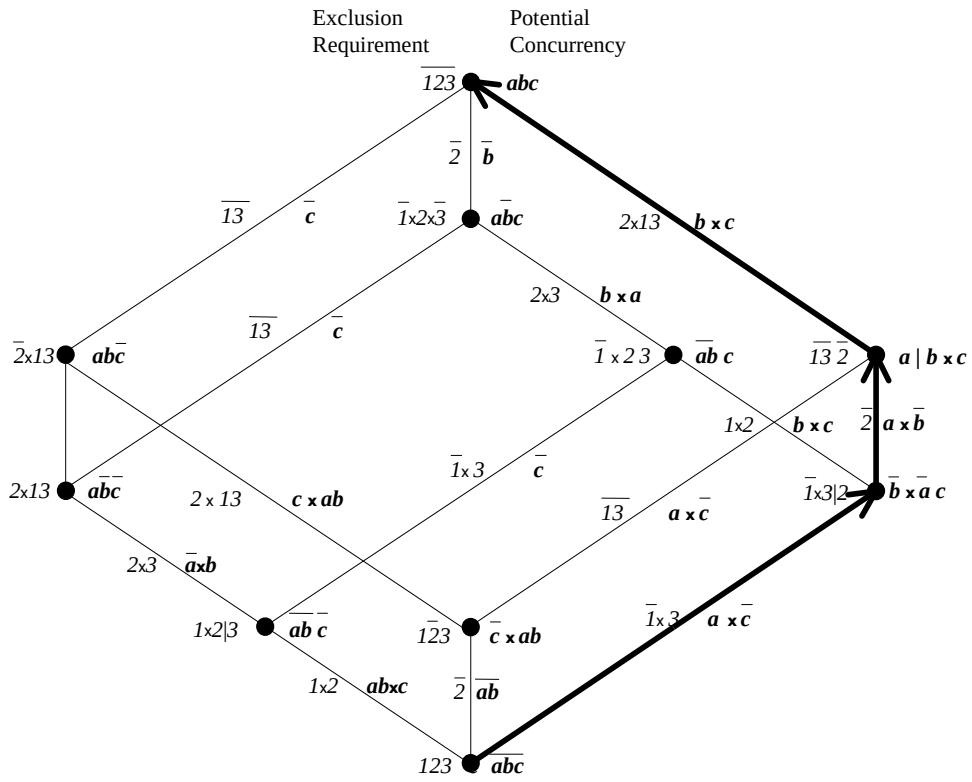


Figure 30: Lock Selections

3.4 Additive Approach - Top-Down Propagation

We now demonstrate the phases of the top-down additive approach for lock selections as we traverse the components top-down. The main idea here is that at any component, if some of the composite exclusion requirements are not met by lock selections at that component, then the remaining requirement should be propagated down to the inner components where these missing requirements are either met or propagated down again, and so on. Any unmet requirements at leaf components must be satisfied by locks there – these locks depend on which locks are selected further out.

Figure 31 shows a composite object with four components. The additive method for lock construction ensure that leaf component are safe at every stage of the lock selection process. In this case, component B and D should always be supplied with minimum locks L_B and L_D to cover their required exclusion R_B and R_D respectively.

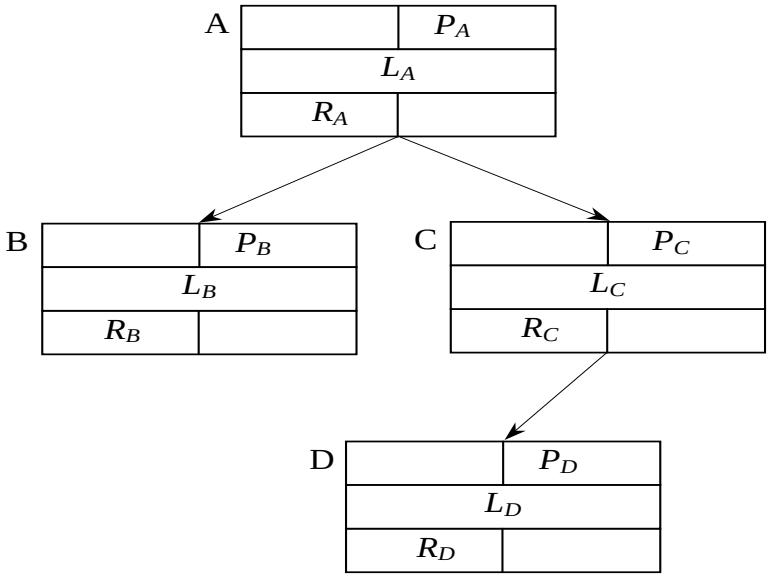


Figure 31: Top Down Example

The following phases summarise the lock selections and calculations for each of the components. Initially we satisfy all requirements at the leaf level, for components B and D:

$$\begin{array}{lllll} Phase_0: & Initialise: & L_A = 0 & & L_C = 0 \\ & Calculate: & L_B = R_B & & L_D = R_D \end{array}$$

Components A and C start with no locks and according to the top-down additive approach, these top components are successively assigned locks that are chosen by users; this decrease the locks that are initially assigned to leaf component as we see in the second phase:

$$\begin{array}{lllll} Phase_1: & Select: & L_A = L_1 & & \\ & Update: & P_B = (P_A - L_1)^{\rightarrow} & P_C = (P_A - L_1)^{\rightarrow} & P_D = P_C^{\rightarrow} \\ & & L_B = R_B \cap P_B & & L_D = R_D \cap P_D \end{array}$$

The new locks for components B and D are automatically calculated by our process which takes into account the new lock placed in the top component A. Component C is assigned a lock in the next phase of our process, that is, after taking into account the lock selected for component A.

$$
\begin{aligned}
Phase_2: \quad Select: \quad & L_C = L_2 \\
Update: \quad & P_D = (P_C - L_2)^{\rightarrow} \\
& L_D = R_D \cap P_D
\end{aligned}
$$

When component C is assigned a lock, the lock of component D is accordingly changed. Locks of B is unchanged.

4 Bottom-Up Approach

A bottom-up approach for offering lock choices can also use either the subtractive or additive method for lock selection. We identify an undesirable characteristic of the subtractive bottom-up approach.

4.1 Sibling Problem for Subtractive Bottom-Up Approach

As with the subtractive lock selection method explained in the top-down approach, we start with a strong lock that meets the minimum exclusion requirements. In this case we initially apply fine-grain locking at the bottom of the composite. Proceeding bottom-up we offer lock choices to be dropped from the inner components, thereby inducing a stronger lock on the outer components. Interestingly this may imply that some locking of siblings of the inner component becomes redundant, and this redundancy should be removed by updating the concurrency potential and induced locks for those components; fortunately this can be done automatically without requiring the user to backtrack in the selection process. However it does mean that some earlier selections for lock removal could be unnecessary because later choices on sibling components would have resulted in their removal anyway. This lack of orthogonality in sibling choices is one reason we prefer the top-down subtractive approach.

4.2 Bottom-Up Additive Approach

In the bottom-up additive approach, we successively add locks bottom-up, and apply any missing requirements with a lock at the outermost level. Initially we start with no internal locking and start adding locks to leaf components. Fortunately the sibling problem identified for the subtractive bottom-up approach does not appear here: later additions of locks on siblings will not make earlier ones redundant.

5 A Model for the Reduced Lattice

In Sections 3.2 and 3.3 we saw that edges in the fixpoint lattice with labels corresponding to given exclusion requirements could be collapsed. In other words, we identify fixpoints $(P_1, R_1), (P_2, R_2)$ when $R \cap R_1 = R \cap R_2$. The idea is that the given requirement R restricts the range of requirements that we are interested in. In Figure 26 we illustrated the reduced lattice, labelling each of the sets of equivalent requirements with the smallest one. Now, rather than showing that the smallest representatives behave consistently with other equivalent elements, we instead reformulate the definition of the fixpoint operator to take the restrictions directly into account. This model will be used in our tool for calculating lock choices in the next section.

We directly generate the reduced lattice (at least an isomorphic variant) by taking into account not only the given inner exclusion requirement, but also the given outer concurrency potential (which we did not restrict in the examples of this report).

In the previous report we defined, for a given usage relation u,

$$\begin{aligned} P^{\rightarrow} &= u^{-1}.P.u \\ P^{\triangleright} &= (P^{\rightarrow})^c \\ R^{\leftarrow} &= u.R.u^{-1} \\ R^{\triangleleft} &= (R^{\leftarrow})^c \end{aligned}$$

Now we simply restrict these operators to the given internal exclusion requirement for the inner component, $R_{I\ inner}$, and the given external concurrency potential for the outer component, $P_{E\ outer}$. So the new definition is:

$$\begin{aligned} P^{\rightarrow} &\widehat{=} R_{I\ inner} \cap u^{-1}.P.u \\ P^{\triangleright} &\widehat{=} R_{I\ inner} - P^{\rightarrow} \\ R^{\leftarrow} &\widehat{=} P_{E\ outer} \cap u.R.u^{-1} \\ R^{\triangleleft} &\widehat{=} P_{E\ outer} - R^{\leftarrow} \end{aligned}$$

As in (UNSW-CSE-TR-604) we can easily prove that these operators $\triangleright$ and $\triangleleft$ do indeed form a Galois connection between outer concurrency potential and inner exclusion requirements. In Figure 32 we show the fixpoint lattice from Figure 26 after restricting all fixpoints (and their differences labelling the edges) to the given exclusion requirement $1 \times \bar{2} \times 3$.

6 Further Properties of the Fixpoint Lattice

For a tool to provide support in the lock selection processes as outlined earlier in this report, it must be able to find the set of fixpoints that cover[1] a given fixpoint. A brute force approach enumerating all elements in the lattice will suffer from exponential complexity in most cases.

In this section we introduce definitions and characterise properties that allow us to

[1] x covers y in a partial order, when $y < x$ and $\forall z \quad y < z \leq x \Rightarrow z = x$.

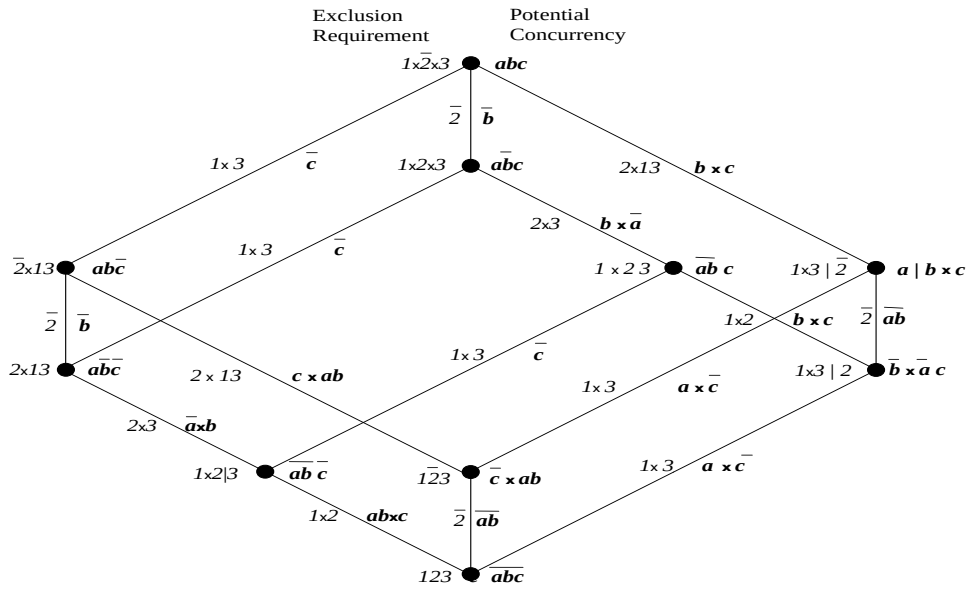


Figure 32: Figure 26 with Fixpoint Restriction

compute the cover of a fixpoint in an efficient way. We then sketch the outline of an algorithm which captures such a computation. This algorithm has been implemented in Haskell to provide us with a simple prototype tool.

In Appendix A we demonstrate the application of this tool to a number of examples, including those of this report. For each example, we display the underlying usage relation, the corresponding lattice (if not too large), and sample interactive traces showing how we can step up the lattice, being offered choices from the covering relation at each step.

Before presenting the details, we refer to the first simple example in Appendix A.1. First, the tool output displays the usage relation and full lattice corresponding to Figure 6.32 (which is repeated in the appendix). After the full lattice, we display the interactive trace for a particular path up through the lattice. This is where the algorithm of this section comes into play. After displaying the current fixpoint the tool offers a number of choices: one for each covering edge of the current fixpoint.

Our algorithm is able to compute these choices efficiently. Essentially the way it does this is to look at each pair p that could be added to the current fixpoint P, and computes

p	$p^{\rightarrow}$	$R \cap p^{\rightarrow}$
$\bar{a}$	$\overline{23}$	2×3
$a \times b$	$\bar{2} \times 3$	2×3
$a \times c$	$1 \times 2 \times \bar{3}$	$1 \times 2 \times 3$
$b \times c$	13×2	13×2
$\bar{c}$	$\overline{13}$	1×3

Table 1: A.1 Example

$(P \cup p)^{\triangleright}$ which is just $R \cap p^{\rightarrow}$. It then groups these, and the minimal groups determine the covering fixpoints $((P \cup p)^{\triangleright\triangleleft}, (P \cup p)^{\triangleright})$. After Step 1 in the example, we find $P = a\bar{b}c$ and $R = 1 \times 2 \times 3$. Then for each pair $p \notin P$, we calculate $p^{\rightarrow}$ and $R \cap p^{\rightarrow}$ as shown in Table 1. In this case, $1 \times 2 \times 3$ and 13×2 are not minimal. We group $\bar{a}$ with $a \times b$ because they yield the same fixpoint. The covering edges are therefore $2 \times 3/\bar{a} \times b$ and $1 \times 3/\bar{c}$, exactly as shown by our tool as increments and decrements in the example of the Appendix.

6.1 Fixpoint Predicate Definitions and Properties

In order to formulate the algorithm for calculating minimal lock choices, we introduce some notions and properties to do with our fixpoint lattice. First we have some definitions dealing with a particular fixpoint R/P. We assume the *Galois* connection $(\triangleright, \triangleleft)$ for the reduced model defined in Section 5.

Definition: Fixpoint

$Fix(P, R) \quad \widehat{=} \quad R = P^{\triangleright} \quad$ **and** $\quad P = R^{\triangleleft}$ □

Definition: Fixpoint above given $Fix(P, R)$

$Up_P(P_1, R_1) \quad \widehat{=} \quad Fix(P_1, R_1) \quad$ **and** $\quad P \subset P_1$ (or equivalently, $R \supset R_1$) □

Definition: Minimum cover above given $Fix(P, R)$

$Min_P(P_1, R_1) \quad \widehat{=} \quad Up_P(P_1, R_1) \quad$ **and** $\quad \forall Up_P(P_2, R_2) \cdot P_2 \subseteq P_1 \Rightarrow P_2 = P_1$

(equivalent: $R_2 \supseteq R_1 \Rightarrow R_2 = R_1$) $\square$

Definition: Difference above given covering $Fix(P, R)$

$Diff_P(\triangle, \nabla) \quad \widehat{=} \quad \triangle \cap P = \phi \quad$ **and** $\quad \nabla \subseteq R \quad$ **and** $\quad Min_P(P \cup \triangle, R - \nabla)$ $\square$

Definition: Fixpoint Difference above given $Fix(P, R)$

Given $X \subseteq P_{E\ outer}$, suppose $Fix(P_1, R_1)$ and P_1 is the smallest fixpoint containing P and X (so, $R_1 = (P \cup X)^{\triangleright}$ and $P_1 = (P \cup X)^{\triangleright\triangleleft}$). The difference to the fixpoint is defined as:

$$\begin{aligned} \nabla_P X &\ \widehat{=}\ R - R_1 \\ \triangle_P X &\ \widehat{=}\ P_1 - P \end{aligned}$$

$\square$

The fixpoints R_1 and P_1 can be determined given these differences. Also we can determine the differences in terms of $X^{\rightarrow}$. We can state these properties precisely.

Properties:

Given the assumptions of the above definition:

$$\begin{aligned} 1.a) \quad R_1 &= R - \nabla_P X \\ 1.b) \quad P_1 &= P \cup \triangle_P X \\ 2.a) \quad \nabla_P X &= R \cap X^{\rightarrow} \\ 2.b) \quad \triangle_P X &= \{p \in P_{E\ outer} - P \mid \nabla_P p \subseteq \nabla_P X\} \end{aligned}$$

Proofs:

1.a) and 1.b) - directly from definition, using $(P \cup X)^{\triangleright} = P^{\triangleright} \cap X^{\triangleright}$.

2.a)

$$
\begin{array}{rcll}
\nabla_P X & = & R - (P \cup X)^{\triangleright} & [\text{ definition of } \nabla \text{ }] \\
& = & R - (R_{I\ inner} - (P \cup X)^{\rightarrow}) & [\text{ definition of } \triangleright \text{ }] \\
& = & R \cap (P \cup X)^{\rightarrow} & [\text{ since } R \subseteq R_{I\ inner} \text{ }] \\
& = & R \cap (P^{\rightarrow} \cup X^{\rightarrow}) & [\text{ since } \rightarrow \text{ distributes over } \cup \\
& & & - \text{ see definition of } \rightarrow \text{ }] \\
& = & R \cap X^{\rightarrow} & [\text{ since } R = P^{\triangleright} = R_{I\ inner} - P^{\rightarrow}, \\
& & & \text{so } R \cap P^{\rightarrow} = \phi \text{ }]
\end{array}
$$

2.b) We prove a more general property:

$$
\begin{array}{rrcll}
 & Y & \subseteq & \triangle_P\ X & \\
\mathsf{iff} & P \cup Y & \subseteq & P \cup \triangle_P\ X & \\
\mathsf{iff} & (P \cup Y)^{\triangleright} & \supseteq & (P \cup \triangle_P\ X)^{\triangleright} & [\ \triangleright \text{ is order reversing }] \\
 & & = & R - \nabla_P X & [\ P \cup \triangle_P\ X \text{ is a fixpoint }] \\
\mathsf{iff} & R - \nabla_P\ Y & \supseteq & R - \nabla_P\ X & [\text{ definition of } \nabla_P \text{ }] \\
\mathsf{iff} & \nabla_P\ Y & \subseteq & \nabla_P\ X & [\text{ both are in } R \text{ }]
\end{array}
$$

□

Now we characterise those X which give us minimal differences. First we introduce notation for describing sets whose elements generate the same differences.

Definition: Equivalent Differences above given $Fix(P, R)$

For each $p \in P_{E\ outer} - P$,

$\equiv_P p \quad \widehat{=} \quad \{x \in P_{E\ outer} - P \mid \nabla_P p = \nabla_P x\}$ □

Properties:

1. $\equiv_P$ just identifies those p which generate the same fixpoint above P.

2. $\{\equiv_P p \mid p \in P_{E\ outer} - P\}$ is a partition of $P_{E\ outer} - P$.

Proof: Obvious from definition of $\equiv_P$. □

There are two key points captured in the following theorem:

a) The minimal differences $(\triangle, \nabla)$ are generated by single elements.

b) The elements which generate minimal differences are those p for which

$$\equiv_P p \quad = \quad \triangle_P p$$

Theorem: Characterisation of Min_P for given $Fix(P, R)$:

$Min_P(\triangle, \nabla)$ iff $\exists\ p \in P_{E\ outer} - P$ s.t. $\nabla = \nabla_P p$ **and** $\triangle\ =\ \equiv_P p\ =\ \triangle_P p$

Proof:

Choose any $p \in \triangle$. By property 2.b above (with $X = \triangle$), it follows that $\nabla\ p\ \subseteq\ \nabla$. But we know:

$$Min_P(\triangle, \nabla) \quad \textbf{and} \quad Up_P(P\ \cup\ \triangle\ p, R - \nabla\ p)$$

By minimality, we must have

$$\nabla\ p \quad = \quad \nabla$$

In other words $\nabla\ p\ =\ \nabla$ for all $p \in \nabla$. The result follows. □

By property 2.b above, it is always true that $x \in \triangle_P\ p \Rightarrow \triangle_P\ x \subseteq \triangle_P\ p$.

The point of the theorem is that minimal differences are characterised by those p such that $\forall\ x \cdot x \in \triangle_P\ p \Rightarrow \triangle_P\ x = \triangle_P\ p$. This forms the basis for the minimal cover algorithm next.

6.2 Sketch of Algorithm

With the definitions and properties outlined above, we can now give the details for the algorithm for computing the cover of a given fixpoint.

Input: u, $P_{E\ outer}$, $R_{I\ inner}$, and fixpoint (P, R)

Output: Differences to covering fixpoints of (P, R)

Steps:

1. For each $p \in P_{E\ outer} - P$, calculate $\nabla_P\ p$ [as $R \cap p^{\rightarrow}$]
2. Partition $P_{E\ outer} - P$ according to $\equiv_P$ (i.e. testing $\nabla_P\ p = \nabla_P\ x$) [$p$ represents its $Part$ of the partition, where $Part = \equiv_P p$ and $\nabla_P Part = \nabla_P p$]
3. Find minimal ∇_P over all parts $\Rightarrow MinPart$

Output: Differences for covers $= \{(Part, \nabla_P\ Part) | Part \in MinPart\}$

Let us apply the algorithm to the example A.5 in the Appendix. We start after step 4; we have a fixpoint with $P = \{$(A,F), (B,F), (C,F), (D,D), (D,E), (D,F), (E,E), (E,F)$\}$, that is, $P = F \times ABC\overline{DE}$. And $R = \{$(1,1), (1,2), (6,6)$\}$, that is, $R = \bar{1} \times 2|\bar{6}$.

$p \in P_{E\ outer} - P$	$p^{\rightarrow}$	Step 1: $\nabla_P p$	Step 2
A,A	1,1	1,1	A,A
A,B	1,1 1,2	1,1 1,2	A,B
A,C	1,1 1,2 1,3	1,1 1,2	A,B
A,D	1,2 1,3 1,4	1,2	A,D
A,E	1,2 1,3 1,4 1,5	1,2	A,D
B,B	1,1 1,2 2,2	1,1 1,2	A,B
B,C	1,1 1,2 1,3 2,2 2,3	1,1 1,2	A,B
B,D	1,2 1,3 1,4 2,2 2,3 2,4	1,2	A,D
B,E	1,2 1,3 1,4 1,5 2,2 2,3 2,4 2,5	1,2	A,D
C,C	1,1 1,2 1,3 2,2 2,3 3,3	1,1 1,2	A,B
C,D	1,2 1,3 1,4 2,2 2,3 2,4 3,3 3,4	1,2	A,D
C,E	1,2 1,3 1,4 1,5 2,2 2,3 2,4 2,5 3,3 3,4 3,5	1,2	A,D
F,F	3,3 3,4 3,5 3,6 4,4 4,5 4,6 5,5 5,6 6,6	6,6	F,F

Table 2: A.5 Example

The first column simply of Table 2 enumerates all those pairs p not in P. The second column takes the image of each of these pairs for the usage relation of A.5. The third column shows the result of Step 1, which calculates $R \cap p^{\rightarrow}$. Column 4 shows the result of Step 2 in which equivalent $p's$ are identified and labelled with one of their representatives. Step 2 involves determining a complete partition of $P_{E\ outer} - P$; results of this are displayed in Table 3.

Part	∇_P
A,A	1,1
A,B A,C B,B B,C C,C	1,1 1,2
A,D A,E B,D B,E C,D C,E	1,2
F,F	6,6

Table 3: Step 2 Completed

Step 3 selects those parts for which ∇_P is minimal. The remaining differences as displayed in Table 4 are identical with the increments and their corresponding decrements that are output by the tool in the appendix. The reader may gain further appreciation of the algorithm by stepping through the other annotated examples in the Appendix.

MinPart	∇_P
A,A	1,1
A,D A,E B,D B,E C,D C,E	1,2
F,F	6,6

Table 4: Step 4

7 Discussion and Conclusion

We have demonstrated the feasibility of automated support for the selection of locks in composite systems. Programmers do not enjoy writing synchronisation code, because it is easy to get wrong. Our approach could be incorporated into development and deployment-time tools, to allow users to select the granularity of synchronisation control in a simple, yet

safe way. The attractiveness of our approach is that we can exploit the Galois connection between concurrency potential and exclusion requirements, to minimise the number of locks that a user needs to choose from, and to order the choices in a structured way.

This report has identified various strategies for lock selection. We observed that the subtractive approach (whether top-down or bottom-up) has the advantage of offering a partition of lock choices at any stage, and users can make their selections by selecting a single representative pair – such choices can be appropriately offered via a 2D matrix display. The additive approach produces overlapping choices, so users must select whole sets of locks at each choice-point rather than single pairs. Furthermore the subtractive approach implies that selections only propagate one level away from current component, so are easier for the user to track.

We have also formulated properties and an algorithm for efficiently calculating the minimal lock choices at any stage of a lock selection process.

References

[SP05] Abdelsalam Shanneb and John Potter. Flexible Exclusion Control for Composite Objects. In *CRPIT '38: Proceedings of the* 28^{th} *Australasian conference on Computer Science*, pages 277–286, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

[SP06] Abdelsalam Shanneb and John Potter. A Galois Connection in Composite Objects. Technical Report -UNSW-CSE-TR-604, Programming Languages and Compilers Group, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, April 2006.

[SPN05] Abdelsalam Shanneb, John Potter, and James Noble. Exclusion Requirements and Potential Concurrency for Composite Objects. *(Elsevier) Science of Computer Programming*, 58(3):344–365, 2005.

A Lock Selection Tool Output

A.1 Lock Selection Tool

In this appendix we display the output of the lock selection tool that was described in the report. We show how the tool displays the set of fixpoints that is associated with the lattice for the examples presented in the report. Also, how the tool steps through the lock selection process.

A.2 Example 1 of Section 3.1

In this example we used the usage relation graph shown in Figure 33, and the given internal exclusion requirement for component B was $R_B = 1 \times \bar{2} \times 3$. The sample run shown below

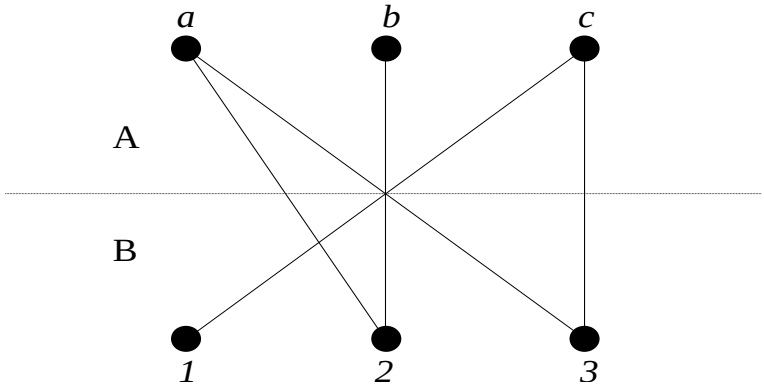


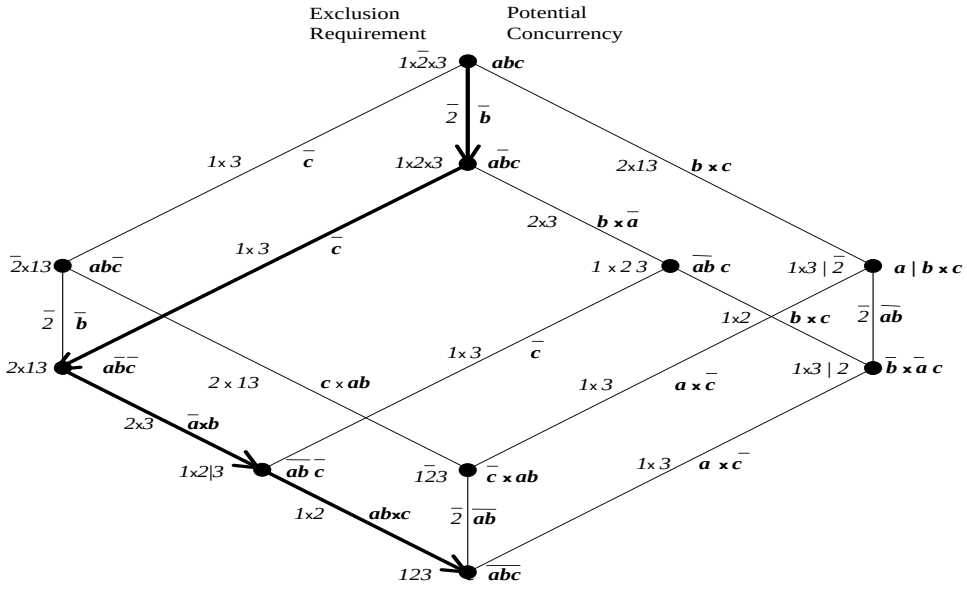
Figure 33: Example 6.3.1 Usage Graph

illustrates the steps of tool output where it first displays all fixpoints of the given usage relation graph and internal exclusion requirement.

```
Relation is:
A  : {2,3}
B  : {2}
C  : {1,3}

Fixpoints:

{}
{(1,2),(1,3),(2,2),(2,3)}

{(B,B)}
{(1,2),(1,3),(2,3)}
```

```
{(A,A),(A,B),(B,B)}
{(1,2),(1,3)}

{(A,A),(A,B),(B,B),(B,C)}
{(1,3)}

{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}
{}

{(A,A),(A,B),(B,B),(C,C)}
{(1,2)}

{(B,B),(C,C)}
{(1,2),(2,3)}

{(B,C)}
{(1,3),(2,2)}

{(A,C),(B,C),(C,C)}
{(2,2)}

{(C,C)}
{(1,2),(2,2),(2,3)}

Number of fixpoints = 10
```

Note that fixpoint values and number agree with the restricted fixpoint lattice of Figure 32 for this example. We show the lattice of fixpoints here again for the sake of ease.



The next part of the tool's output is to step down the fixpoint lattice. The user is asked to choose a step from the current fixpoint. We follow the same path taken in example 6.1, so we first chose path or option 1 which leads to fixpoint $\bar{1} \times 2 \times \bar{3}$. You can easily follow the rest of the choices the tools displays off the shown lattice.

```
Fixpoint is:

{}
{(1,2),(1,3),(2,2),(2,3)}

Increments (should be unique):
  #1 : {(B,B)}
  #2 : {(C,C)}
  #3 : {(B,C)}

Corresponding decrements (may overlap):
  #1 : {(2,2)}
  #2 : {(1,3)}
  #3 : {(1,2),(2,3)}
--------------------------------------------------
Step 1.

Choose: #1
  +=   {(B,B)}
  -=   {(2,2)}
--------------------------------------------------
Fixpoint is:

{(B,B)}
{(1,2),(1,3),(2,3)}

Increments (should be unique):
  #1 : {(A,A),(A,B)}
  #2 : {(C,C)}

Corresponding decrements (may overlap):
  #1 : {(2,3)}
  #2 : {(1,3)}
--------------------------------------------------
Step 2.

Choose: #2
  +=   {(C,C)}
  -=   {(1,3)}
--------------------------------------------------
Fixpoint is:

{(B,B),(C,C)}
{(1,2),(2,3)}

Increments (should be unique):
  #1 : {(A,A),(A,B)}

Corresponding decrements (may overlap):
  #1 : {(2,3)}
--------------------------------------------------
Step 3.

Choose: #1
  +=   {(A,A),(A,B)}
  -=   {(2,3)}
--------------------------------------------------
Fixpoint is:

{(A,A),(A,B),(B,B),(C,C)}
{(1,2)}

Increments (should be unique):
  #1 : {(A,C),(B,C)}

Corresponding decrements (may overlap):
  #1 : {(1,2)}
--------------------------------------------------
Step 4.

Choose: #1
  +=   {(A,C),(B,C)}
  -=   {(1,2)}
--------------------------------------------------
Fixpoint is:
```

```
{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}
{}

No more increments.
```

A.3 Example 2 of Section 3.2

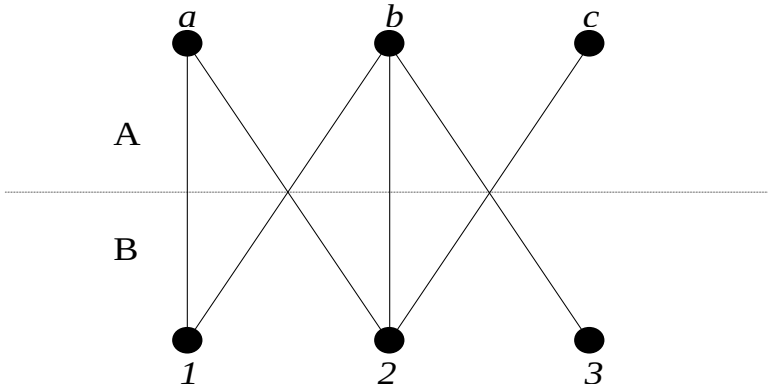


Figure 34: Example 6.3.2 Usage Graph

In this example we used the usage relation graph of Example 2 of Section 3.2, as shown in Figure 34. The tool displays the calculated fixpoints as shown below:

```
Relation is:
A  : {1,2}
B  : {1,2,3}
C  : {2}

Fixpoints:

{}
{(1,2),(1,3),(2,2),(2,3)}

{(C,C)}
{(1,2),(1,3),(2,3)}

{(A,A),(A,C),(C,C)}
{(1,3),(2,3)}

{(A,A),(A,C),(B,C),(C,C)}
{(1,3)}

{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}
{}

Number of fixpoints = 5
```

Under the given internal exclusion requirement $R_B = 1 \times \bar{2} \times 3$, we saw in Section 3.2 that the final fixpoint lattice formed had five fixpoints as in Figure 2.1. Our tool sees the fixpoints of the isomorphic reduced lattice, as described in Section 5 — we show the corresponding diagram in Figure 35. This lattice has only one path for lock selection as shown in the output steps of the tool.

```
Fixpoint is:
```

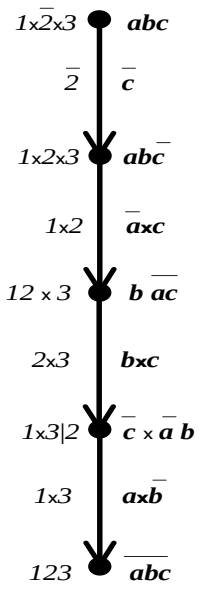


Figure 35: Example 6.3.2 Lattice

```
{}
{(1,2),(1,3),(2,2),(2,3)}

Increments (should be unique):
  #1 : {(C,C)}

Corresponding decrements (may overlap):
  #1 : {(2,2)}

--------------------------------------------------
Step 1.

Choose: #1

  +=   {(C,C)}
  -=   {(2,2)}
--------------------------------------------------
Fixpoint is:

{(C,C)}
{(1,2),(1,3),(2,3)}


Increments (should be unique):
  #1 : {(A,A),(A,C)}

Corresponding decrements (may overlap):
  #1 : {(1,2)}

--------------------------------------------------
Step 2.

Choose: #1

  +=   {(A,A),(A,C)}
  -=   {(1,2)}
--------------------------------------------------
Fixpoint is:

{(A,A),(A,C),(C,C)}
{(1,3),(2,3)}


Increments (should be unique):
```

```
  #1 : {(B,C)}

Corresponding decrements (may overlap):
  #1 : {(2,3)}

------------------------------------------------
Step 3.

Choose: #1

  +=   {(B,C)}
  -=   {(2,3)}
------------------------------------------------
Fixpoint is:

{(A,A),(A,C),(B,C),(C,C)}
{(1,3)}

Increments (should be unique):
  #1 : {(A,B),(B,B)}

Corresponding decrements (may overlap):
  #1 : {(1,3)}

------------------------------------------------
Step 4.

Choose: #1

  +=   {(A,B),(B,B)}
  -=   {(1,3)}
------------------------------------------------
Fixpoint is:

{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}
{}

No more increments.
```

A.4 Example 3 of Section 3.3

Example 3 of Section 3.3 illustrates the choices for adding to the outer component. In particular note the overlap in the choices for Step 1 where (A,C) appears in both choice #1 and #2. This overlap corresponds to the question mark in matrix (1) if Figure 27.

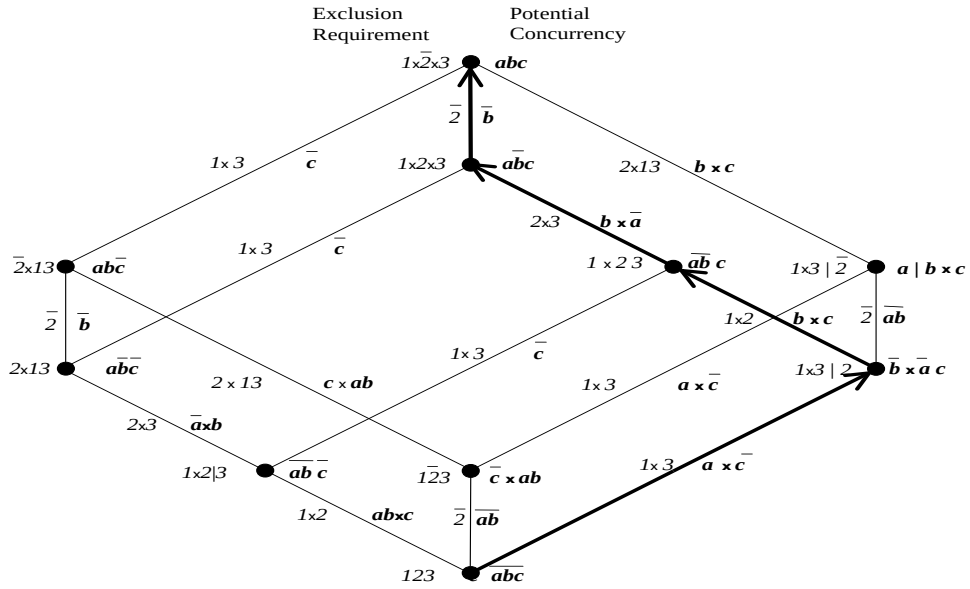


Figure 36: Example 6.3.3 Lattice

```
Relation is:
1  : {C}
2  : {A,B}
3  : {A,C}

Fixpoints:

{}
{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}

{(1,2)}
{(A,A),(A,B),(B,B),(C,C)}

{(1,2),(1,3)}
{(A,A),(A,B),(B,B)}

{(1,2),(1,3),(2,3)}
{(B,B)}

{(1,2),(1,3),(2,2),(2,3)}
{}

{(1,2),(2,3)}
{(B,B),(C,C)}

{(1,2),(2,2),(2,3)}
{(C,C)}

{(1,3)}
{(A,A),(A,B),(B,B),(B,C)}

{(1,3),(2,2)}
```

```
{(B,C)}

{(2,2)}
{(A,C),(B,C),(C,C)}


Number of fixpoints = 10

Fixpoint is:

{}
{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}


Increments (should be unique):
  #1 : {(1,2)}
  #2 : {(1,3)}
  #3 : {(2,2)}

Corresponding decrements (may overlap):
  #1 : {(A,C),(B,C)}
  #2 : {(A,C),(C,C)}
  #3 : {(A,A),(A,B),(B,B)}

------------------------------------------------
Step 1.

Choose: #2

  +=   {(1,3)}
  -=   {(A,C),(C,C)}
------------------------------------------------
Fixpoint is:

{(1,3)}
{(A,A),(A,B),(B,B),(B,C)}


Increments (should be unique):
  #1 : {(1,2)}
  #2 : {(2,2)}

Corresponding decrements (may overlap):
  #1 : {(B,C)}
  #2 : {(A,A),(A,B),(B,B)}

------------------------------------------------
Step 2.

Choose: #1

  +=   {(1,2)}
  -=   {(B,C)}
------------------------------------------------
Fixpoint is:

{(1,2),(1,3)}
{(A,A),(A,B),(B,B)}


Increments (should be unique):
  #1 : {(2,3)}

Corresponding decrements (may overlap):
  #1 : {(A,A),(A,B)}

------------------------------------------------
Step 3.

Choose: #1

  +=   {(2,3)}
  -=   {(A,A),(A,B)}
------------------------------------------------
Fixpoint is:
```

```
{(1,2),(1,3),(2,3)}
{(B,B)}


Increments (should be unique):
  #1 : {(2,2)}

Corresponding decrements (may overlap):
  #1 : {(B,B)}

------------------------------------------------
Step 4.

Choose: #1

  +=   {(2,2)}
  -=   {(B,B)}
------------------------------------------------
Fixpoint is:

{(1,2),(1,3),(2,2),(2,3)}
{}


No more increments.
```

A.5 A Large Example

Let's try a larger set of methods. In this example we consider a component with 5 internal methods and the usage graph as shown in Figure 37. From the usage graph it is clear that no exclusion requirements are needed since all methods are independent.

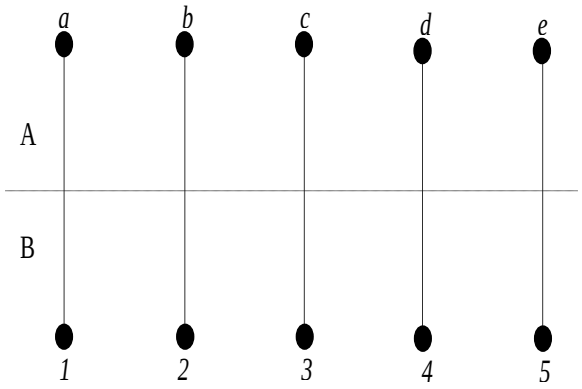


Figure 37: Usage Graph of a Large Component

The number of elements for this fixpoint lattice will be 32768 according to the formula $2^{\frac{n(n+1)}{2}}$, where n is the number of elements. We only show the first few lines and the last lines of the output.

```
Relation is:

A  : {1}
B  : {2}
C  : {3}
D  : {4}
E  : {5}

Fixpoints:

{}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),
(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A)}
{(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B)}
{(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),
(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C)}
{(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),
(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D)}
{(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E)}
```

```
{(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B)}
{(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C)}
{(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D)}
{(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E)}
{(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C)}
{(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D)}
{(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E)}
{(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(D,D)}
{(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(D,D),(D,E)}
{(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(D,D),(D,E),(E,E)}
{}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(D,D),(E,E)}
{(4,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(D,E)}
{(4,4),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(D,E),(E,E)}
{(4,4)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(C,E),(E,E)}
{(4,4),(4,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(D,D)}
{(3,5),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(D,D),(D,E)}
{(3,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(D,D),(D,E),(E,E)}
{(3,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),
(D,D),(E,E)}
{(3,5),(4,5)}

.
.
.
.
.
.
.
.
.
```

```
{(C,E),(D,E),(E,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(4,4)}

{(C,E),(E,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(4,4),(4,5)}

{(D,D)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(4,5),(5,5)}

{(D,D),(D,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(5,5)}

{(D,D),(D,E),(E,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5)}

{(D,D),(E,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(4,5)}

{(D,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(4,4),(5,5)}

{(D,E),(E,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(4,4)}

{(E,E)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),
(3,5),(4,4),(4,5)}

Number of fixpoints = 32768
```

For the same set of methods and for a different usage relation graph we get a smaller size fixpoint lattice.

```
Relation is:

A  : {1}
B  : {1,2}
C  : {1,2,3}
D  : {1,2,3,4}
E  : {1,2,3,4,5}

Fixpoints:

{}
{(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),
(4,4),(4,5),(5,5)}

{(A,A)}
{(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),
(4,5),(5,5)}

{(A,A),(A,B)}
{(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),
(5,5)}

{(A,A),(A,B),(A,C)}
{(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D)}
{(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E)}
{(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}
```

```
{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B)}
{(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C)}
{(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D)}
{(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E)}
{(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C)}
{(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D)}
{(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),(C,E)}
{(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),(C,E),
(D,D)}
{(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),(C,E),
(D,D),(D,E)}
{(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),(C,E),
(D,D),(D,E),(E,E)}
{}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(B,E),(C,C),(C,D),(D,D)}
{(3,5),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(C,C)}
{(2,5),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(C,C),(C,D)}
{(2,5),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(B,D),(C,C),(C,D),(D,D)}
{(2,5),(3,5),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(A,E),(B,B),(B,C),(C,C)}
{(2,4),(2,5),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B)}
{(1,5),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C)}
{(1,5),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D)}
{(1,5),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D),(C,C)}
{(1,5),(2,5),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D),(C,C),(C,D)}
{(1,5),(2,5),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D),(C,C),(C,D),(D,D)}
{(1,5),(2,5),(3,5),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(C,C)}
{(1,5),(2,4),(2,5),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(B,B)}
{(1,4),(1,5),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(B,B),(B,C)}
{(1,4),(1,5),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

{(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}
{(1,4),(1,5),(2,4),(2,5),(3,4),(3,5),(4,4),(4,5),(5,5)}
```

```
{(A,A),(A,B),(B,B)}
{(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5),(4,4),(4,5),(5,5)}

Number of fixpoints = 32
```

A.6 A Short Path Example

In this example we demonstrate how a lock selection may take just few steps to secure a useful lock distribution even with large interfaces. The given usage relation and part of produced fixpoints are shown below:

```
Relation is:
A  : {1}
B  : {1,2}
C  : {1,2,3}
D  : {2,3,4}
E  : {2,3,4,5}
F  : {3,4,5,6}

Fixpoints:

{}
{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),
(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

{(A,A)}
{(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),(3,3),
(3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

{(A,A),(A,B)}
{(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),
(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

{(A,A),(A,B),(A,C)}
{(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),(3,5),
(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

{(A,A),(A,B),(A,C),(A,D)}
{(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),(3,5),(3,6),
(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

.
.
.


{(D,F),(E,F),(F,F)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2)}

{(F,F)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6)}

Number of fixpoints = 270
```

We now show the chosen steps:

```
Fixpoint is:

{}
{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),
(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

Increments (should be unique):
  #1 : {(A,A)}
  #2 : {(A,D)}
  #3 : {(A,F)}
```

```
  #4 : {(D,D)}
  #5 : {(F,F)}
  #6 : {(D,F)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}
  #2 : {(1,2),(1,3),(1,4)}
  #3 : {(1,3),(1,4),(1,5),(1,6)}
  #4 : {(2,2),(2,3),(2,4),(3,3),(3,4),(4,4)}
  #5 : {(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}
  #6 : {(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6)}

--------------------------------------------------
Step 1.

Choose: #6

  +=   {(D,F)}
  -=   {(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6)}
--------------------------------------------------
Fixpoint is:

{(D,F)}
{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(5,5),(5,6),(6,6)}


Increments (should be unique):
  #1 : {(A,A)}
  #2 : {(D,D),(D,E)}
  #3 : {(E,F)}
  #4 : {(A,D)}
  #5 : {(A,F),(B,F),(C,F)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}
  #2 : {(2,2)}
  #3 : {(5,5),(5,6)}
  #4 : {(1,2),(1,3),(1,4)}
  #5 : {(1,3),(1,4),(1,5),(1,6)}

--------------------------------------------------
Step 2.

Choose: #5

  +=   {(A,F),(B,F),(C,F)}
  -=   {(1,3),(1,4),(1,5),(1,6)}
--------------------------------------------------
Fixpoint is:

{(A,F),(B,F),(C,F),(D,F)}
{(1,1),(1,2),(2,2),(5,5),(5,6),(6,6)}


Increments (should be unique):
  #1 : {(A,A)}
  #2 : {(A,D),(A,E)}
  #3 : {(D,D),(D,E)}
  #4 : {(E,F)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}
  #2 : {(1,2)}
  #3 : {(2,2)}
  #4 : {(5,5),(5,6)}

--------------------------------------------------
Step 3.

Choose: #4

  +=   {(E,F)}
  -=   {(5,5),(5,6)}
--------------------------------------------------
Fixpoint is:
```

```
{(A,F),(B,F),(C,F),(D,F),(E,F)}
{(1,1),(1,2),(2,2),(6,6)}


Increments (should be unique):
  #1 : {(A,A)}
  #2 : {(A,D),(A,E)}
  #3 : {(D,D),(D,E),(E,E)}
  #4 : {(F,F)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}
  #2 : {(1,2)}
  #3 : {(2,2)}
  #4 : {(6,6)}

------------------------------------------------
Step 4.

Choose: #3

  +=   {(D,D),(D,E),(E,E)}
  -=   {(2,2)}
------------------------------------------------
Fixpoint is:

{(A,F),(B,F),(C,F),(D,D),(D,E),(D,F),(E,E),(E,F)}
{(1,1),(1,2),(6,6)}


Increments (should be unique):
  #1 : {(A,A)}
  #2 : {(A,D),(A,E),(B,D),(B,E),(C,D),(C,E)}
  #3 : {(F,F)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}
  #2 : {(1,2)}
  #3 : {(6,6)}

------------------------------------------------
Step 5.

Choose: #3

  +=   {(F,F)}
  -=   {(6,6)}
------------------------------------------------
Fixpoint is:

{(A,F),(B,F),(C,F),(D,D),(D,E),(D,F),(E,E),(E,F),(F,F)}
{(1,1),(1,2)}


Increments (should be unique):
  #1 : {(A,A)}
  #2 : {(A,D),(A,E),(B,D),(B,E),(C,D),(C,E)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}
  #2 : {(1,2)}

------------------------------------------------
Step 6.

Choose: #2

  +=   {(A,D),(A,E),(B,D),(B,E),(C,D),(C,E)}
  -=   {(1,2)}
------------------------------------------------
Fixpoint is:

{(A,D),(A,E),(A,F),(B,D),(B,E),(B,F),(C,D),(C,E),(C,F),(D,D),(D,E),
(D,F),(E,E),(E,F),(F,F)}
{(1,1)}
```

```
Increments (should be unique):
  #1 : {(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}

Corresponding decrements (may overlap):
  #1 : {(1,1)}

--------------------------------------------------
Step 7.

Choose: #1

  +=   {(A,A),(A,B),(A,C),(B,B),(B,C),(C,C)}
  -=   {(1,1)}
--------------------------------------------------
Fixpoint is:

{(A,A),(A,B),(A,C),(A,D),(A,E),(A,F),(B,B),(B,C),(B,D),(B,E),(B,F),
(C,C),(C,D),(C,E),(C,F),(D,D),(D,E),(D,F),(E,E),(E,F),(F,F)}
{}


No more increments.
```

A.7 An Example with Ambiguity

This example illustrates some of the theoretical properties outlined in Section 6. The increments are all non-overlapping, whereas the decrements may overlap arbitrarily. Consider the usage relations shown below:

```
Relation is:
A  : {1,2}
B  : {2,4}
C  : {3}
D  : {1,4}
```

Using the inner exclusion requirement $R_B = \overline{1234}$, we get 140 fixpoints, some of which we list below:

```
Fixpoints:

{}
{(1,1),(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,3),(3,4),(4,4)}

{(A,A)}
{(1,3),(1,4),(2,3),(2,4),(3,3),(3,4),(4,4)}

{(A,A),(A,B),(A,D)}
{(1,3),(2,3),(3,3),(3,4),(4,4)}

{(A,A),(A,B),(A,C),(A,D)}
{(3,3),(3,4),(4,4)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,D),(D,D)}
{(3,3),(3,4)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D),(C,D),(D,D)}
{(3,3)}

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D),(C,C),(C,D),(D,D)}
{}

.
.
.
.

{(C,C)}
{(1,1),(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,4),(4,4)}

{(C,C),(C,D)}
{(1,1),(1,2),(1,4),(2,2),(2,3),(2,4),(4,4)}

{(C,C),(C,D),(D,D)}
{(1,2),(2,2),(2,3),(2,4)}

{(C,C),(D,D)}
{(1,2),(1,3),(2,2),(2,3),(2,4),(3,4)}

{(C,D)}
{(1,1),(1,2),(1,4),(2,2),(2,3),(2,4),(3,3),(4,4)}

{(C,D),(D,D)}
{(1,2),(2,2),(2,3),(2,4),(3,3)}

{(D,D)}
{(1,2),(1,3),(2,2),(2,3),(2,4),(3,3),(3,4)}

Number of fixpoints = 140
```

Let's examine the first set of lock choices:

```
Fixpoint is:

{}
{(1,1),(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,3),(3,4),(4,4)}


Increments (should be unique):
  #1 : {(C,C)}
  #2 : {(A,C)}
  #3 : {(B,C)}
  #4 : {(C,D)}
  #5 : {(A,A)}
  #6 : {(B,B)}
  #7 : {(D,D)}
  #8 : {(A,B)}
  #9 : {(A,D)}
  #10: {(B,D)}

Corresponding decrements (may overlap):
  #1 : {(3,3)}
  #2 : {(1,3),(2,3)}
  #3 : {(2,3),(3,4)}
  #4 : {(1,3),(3,4)}
  #5 : {(1,1),(1,2),(2,2)}
  #6 : {(2,2),(2,4),(4,4)}
  #7 : {(1,1),(1,4),(4,4)}
  #8 : {(1,2),(1,4),(2,2),(2,4)}
  #9 : {(1,1),(1,2),(1,4),(2,4)}
  #10: {(1,2),(1,4),(2,4),(4,4)}
```

We can clearly see some redundancy in the choices for decrements. Choice #5 is replicated in other choices, so we can not identify it by a single pair (m, n). We continue our selection of offered locks:

```
--------------------------------------------------
Step 1.

Choose: #7

  +=   {(D,D)}
  -=   {(1,1),(1,4),(4,4)}
--------------------------------------------------
Fixpoint is:

{(D,D)}
{(1,2),(1,3),(2,2),(2,3),(2,4),(3,3),(3,4)}


Increments (should be unique):
  #1 : {(C,C)}
  #2 : {(A,A)}
  #3 : {(A,C)}
  #4 : {(A,D),(B,D)}
  #5 : {(B,B)}
  #6 : {(B,C)}
  #7 : {(C,D)}

Corresponding decrements (may overlap):
  #1 : {(3,3)}
  #2 : {(1,2),(2,2)}
  #3 : {(1,3),(2,3)}
  #4 : {(1,2),(2,4)}
  #5 : {(2,2),(2,4)}
  #6 : {(2,3),(3,4)}
  #7 : {(1,3),(3,4)}
```

```
------------------------------------------------
Step 2.

Choose: #3

  +=   {(A,C)}
  -=   {(1,3),(2,3)}
------------------------------------------------
Fixpoint is:

{(A,C),(D,D)}
{(1,2),(2,2),(2,4),(3,3),(3,4)}


Increments (should be unique):
  #1 : {(B,C),(C,D)}
  #2 : {(C,C)}
  #3 : {(A,A)}
  #4 : {(A,D),(B,D)}
  #5 : {(B,B)}

Corresponding decrements (may overlap):
  #1 : {(3,4)}
  #2 : {(3,3)}
  #3 : {(1,2),(2,2)}
  #4 : {(1,2),(2,4)}
  #5 : {(2,2),(2,4)}

------------------------------------------------
Step 3.

Choose: #4

  +=   {(A,D),(B,D)}
  -=   {(1,2),(2,4)}
------------------------------------------------
Fixpoint is:

{(A,C),(A,D),(B,D),(D,D)}
{(2,2),(3,3),(3,4)}


Increments (should be unique):
  #1 : {(A,A),(A,B),(B,B)}
  #2 : {(B,C),(C,D)}
  #3 : {(C,C)}

Corresponding decrements (may overlap):
  #1 : {(2,2)}
  #2 : {(3,4)}
  #3 : {(3,3)}

------------------------------------------------
Step 4.

Choose: #3

  +=   {(C,C)}
  -=   {(3,3)}
------------------------------------------------
Fixpoint is:

{(A,C),(A,D),(B,D),(C,C),(D,D)}
{(2,2),(3,4)}


Increments (should be unique):
  #1 : {(A,A),(A,B),(B,B)}
  #2 : {(B,C),(C,D)}

Corresponding decrements (may overlap):
  #1 : {(2,2)}
  #2 : {(3,4)}
```

```
------------------------------------------------
Step 5.

Choose: #2

  +=   {(B,C),(C,D)}
  -=   {(3,4)}
------------------------------------------------
Fixpoint is:

{(A,C),(A,D),(B,C),(B,D),(C,C),(C,D),(D,D)}
{(2,2)}


Increments (should be unique):
  #1 : {(A,A),(A,B),(B,B)}

Corresponding decrements (may overlap):
  #1 : {(2,2)}

------------------------------------------------
Step 6.

Choose: #1

  +=   {(A,A),(A,B),(B,B)}
  -=   {(2,2)}
------------------------------------------------
Fixpoint is:

{(A,A),(A,B),(A,C),(A,D),(B,B),(B,C),(B,D),(C,C),(C,D),(D,D)}
{}


No more increments.
```